A VERIFICATION SYSTEM FOR NC PROGRAMMING FOR QUALITY IN MILLING

By

MAHESH NALLAKATLA

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1992

ACKNOWLEDGEMENTS

The author would like to express his sincere gratitude to Dr. Jiri Tlusty and Dr. Scott Smith for their guidance and generous support during this research and for giving him the opportunity to work in the Machine Tool Laboratory. The author also thanks Dr. Carl Crane, Dr. John Schueller and Dr. Sencer Yeralan for serving on his committee. Special thanks go to present and past members of the Machine Tool Laboratory with whom the author shared many experiences that made a significant contribution towards his graduate education. He is specially thankful to Dr. E. Cheng, Dr. R.L. Wells, J. Frost, Dr. C. Zamudio, and C. Vierck for their help and constructive evaluations during the critical stages of the research. The author also wishes to thank Dr. A. DeVries from the Electrical Engineering Department for contributing his expertise to the project.

TABLE OF CONTENTS

page

ACKNOWLEDGEMENTS ii

ABSTRACT v

CHAPTERS

1 INTRODUCTION 1

Programming for Quality in Milling 1
Justification for a Verification System 5
Chatter Considerations and Limits of Stability 7
Review of Literature 12
Outline of the Present Research 20

2 WORKPIECE AND TOOL SWEPT AREA REPRESENTATIONS 22

Wireframe Models 22
Workpiece Representation 24
Tool Swept Area Representation 28

3 BOOLEAN OPERATIONS 35

Outline of Algorithms 35
Determination of Intersection Points 36
Difference and Intersection Operations 40

4 VERIFICATION TECHNIQUES 48

Extraction of Cutting Parameters 49
Monitoring the Limits of Stability 61
Reprogramming of the NC file 69

5 MODIFICATIONS IN THE REPRESENTATION SCHEME 72

Dynamic Spiral-in Pocketing Routine 72
The Modified Representation Scheme 80

6 APPLICATIONS IN COMMERCIAL POCKETING ROUTINES 87

Description of the Pocketing Routines 87
Spiral-out Pockets without Islands 90

Spiral-out Pockets with Islands 95
Zig-zag Pockets with and without Islands 100
A Specialized Pocketing Routine 108

7 CONCLUSIONS AND RECOMMENDATIONS 112

APPENDIX
SUPPORT ALGORITHMS FOR THE BOOLEAN OPERATIONS 115

Intersection Between Two Lines 115
Intersection Between a Line and an Arc 116
Intersection Between two Arcs 118
Check for Containment of a Point on a Line 119
Check for Containment of a Point on an Arc 120
Check for Containment of a Point Between Two Points . 121

REFERENCES 122

BIOGRAPHICAL SKETCH 125

Abstract of Dissertation Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

A VERIFICATION SYSTEM FOR NC PROGRAMMING FOR QUALITY IN MILLING

By

Mahesh Nallakatla

August 1992

Chairman: Dr. Jiri Tlusty
Major Department: Mechanical Engineering

Most of the previous work in the area of verification systems has concentrated on aspects such as gouging during the cutting operation, tool-fixture collisions and detection of excess material that may have been left behind. The prediction of chatter which plays a very important role in reducing machining overheads and in increasing metal removal rates has not been substantially considered in currently available commercial systems. This dissertation outlines the development of a verification system to examine Numerical Control (NC) programs and to predict the occurrence of chatter in common milling operations such as pocketing.

This system plays the role of an off-line tool to ensure chatter-free machining and increased metal removal rates. The NC program, which contains a list of the various tool

movements, does not provide explicit information about cutting parameters such as the radial and axial depths of cut, feed directions, or the mode of milling (up or down milling), which are needed for chatter prediction. It is therefore necessary to extract these parameters from models which utilize the tool trajectories contained in the NC program.

Algorithms are presented which refer to previously established databases of the limiting values of axial and radial depths for chatter avoidance. Additionally, certain corrective actions that are undertaken when the extracted values exceed these limit values are outlined. The pockets considered are restricted to simple 2 1/2 dimensional geometries and may include internal islands. Several examples are used to illustrate the application of the above system using pocketing NC programs generated using commercial NC packages and using some special algorithms.

CHAPTER 1
INTRODUCTION

The various issues that motivate the development of a verification system for stability in Numerical Control machining are discussed in this chapter. A review of the research in the areas of Solid Modeling and the extraction of cutter engagements from Computer Numerical Control (CNC) tool trajectories is presented. The guidelines used to guarantee a high degree of reliability in the outcome of machining operations are emphasized. An outline of the present research in developing an off-line verification system to monitor the cutter engagements is also provided.

Programming for Quality in Milling

The milling operation which has been investigated in this work is the pocketing routine. A pocket is defined by its outside boundaries and internal island boundaries if there exists a profile within the pocket. Pocketing is generally used in die sinking and machining of airframe panels. The quality of a milling operation is quantified by four parameters: dimensional error e_d, profile error e_p, surface

error e_s, and the amplitude of the variable component of the cutting force F_v (Smith and Tlusty, 1988). Modification of existing pocketing routines and the development of new routines have helped to provide a set of rules for determining the parameters used in these routines (like the offsets of passes and depth of passes) which will satisfy the quality characteristics specified by the above four parameters. The effects of radial depth a, and axial depth b, are emphasized in these techniques in contrast to work in the past which concentrated on the cutting speed v, and feed f. While changing feeds f and speeds v does not affect the geometry of the part or the Numerical Control (NC) program (M and G code file), changes of a and b result in drastic NC program changes. Additionally it should be pointed out here that the radial and axial depths are the parameters which are most decisive in chatter free machining and increased MRR (Tlusty and Smith, 1985).

In commercial NC programming packages, the programmer is left with only a general recommendation for the depth of cut value, such as a = 0.75d for roughing, and a = 0.2d for finishing, where d is the tool diameter (General Electric, 1980). Additional guidelines for the selection of cutting speed and chip load are based on an extensive amount of accumulated empirical data (e.g. Metcut, 1972). The other group of parameters defining the geometry of the cut, the radial depth of cut, the axial depth of cut and also the

number of teeth have been left to the programmer's discretion. Because of this, the cycle of tool path programming and trial cuts maybe repeated several times before a part with satisfactory quality parameters is achieved.

The above parameters which constitute the inputs to the pocketing routines can be obtained by either computational techniques or by well developed cutting tests. Zamudio (1991) has developed a methodology for the complete characterization of machine tools which includes the computation and measurement of chatter limits and errors of accuracy. These results can be used by the part programmer to optimally select cutting parameters which will result in stable cutting in the roughing cuts and in minimized errors in the finishing cuts. In his methodology, the measurement of transfer functions and computation of chatter limits is verified by cutting tests for a selection of tools.

The most precise and reliable method for the computation of chatter limits is the time domain simulation of the milling process (Tlusty and Ismail, 1983). The algorithm is based on the regeneration of waviness phenomenon, where the waviness imprinted on the surface as the teeth pass combines with the motion of the cutting teeth to produce chip thickness variations. This chip thickness variation results in a variable cutting force which feeds the vibration. Frequency domain records of the cutting force and the vibration of the

system are obtained and examined to determine whether the system is stable or unstable.

The commercial pocketing routines provide either linear up/down or spiral inside/outside tool paths. It is assumed in all instances that the axial and radial cutter engagements remain constant throughout the pocket. The milling machine dynamics are not considered in the generation of the tool path. Kshonze (1989) has developed dynamic pocketing routines which consider the machine dynamics and the mode of milling in the design of the tool path. An additional feature in these routines is the ability to specify different axial depths of cut for different radial immersions, while still maintaining a stable cut throughout the machining process. The problems of increased radial immersions at internal corners, and plunging of the tool into the workpiece at the beginning of each pass in the zig-zag pocketing option, are also effectively dealt with in these routines.

Cheng (1991) has modified the dynamic spiral pocketing routine and has developed techniques for the generation of tool paths for any two-and-a-half-dimensional pockets with islands. The database for this routine (containing the permissible axial and radial depths) is calculated using the results obtained from the characterization of the machine tool (Zamudio 1991). The tool path generation algorithm utilizes a binary tree representation for the pocket and the spiral tool paths. The tool path is generated via an external program

(written in C) and then passed on to any Computer Aided Design/Manufacturing (CAD/CAM) system for viewing. The CAD/CAM package currently used for this purpose is CATIA. CATIA has a degree of flexibility in that it provides complete access to the user, from the stage of part design to the generation of NC data. Current work is being concentrated on extending the application of this routine on commercial PC-based packages such as SMARTCAM and EXPEDITE-3D. DXF files which are commonly used in AUTOCAD, are used for the interchange of data.

Justification for a Verification System

The above mentioned approaches are geared towards generating NC programs which are chatter free. Kshonze's dynamic pocketing routines involve manual programming of the original tool path. This is necessary due to the inaccessibility of the source code in the commercial packages. Manual programming of the tool paths is a very time consuming process and is generally not recommended for complicated pocket geometries, especially when islands are involved. The above problems are effectively dealt with in Cheng's approach, where the tool path is generated in external software. Different CAD/CAM packages have varying modes of interaction with external programs. CATIA communicates with external programs via an Interactive User Access (IUA) facility, while SMARTCAM utilizes it's own CAM communication module. The task

of communicating with these packages would be significantly easier if a standard data transfer protocol were used, which would eliminate the extra effort in developing a processing module for each of these packages to facilitate interaction.

Kshonze's and Cheng's approach can be regarded as a front end optimization process, since the task of chatter control is being carried out during the actual tool path generation procedure. The axial and radial depths are programmed into the routine, after being predetermined from a database created from the characterization of the machine tool.

The present research has concentrated on developing a verification system which could be used to examine existing NC code for common milling operations such as pocketing and contouring. This is a monitoring system which checks that the cutting parameters do not exceed the limits of stability. Each block of code in the NC program is scanned to extract the cutter engagements, feed direction and the mode of milling, and these values are checked against the limit values stored in a database similar to the one used in Cheng's approach. A complete outline of this system is provided later on in this chapter.

The above mentioned system can be regarded as a viable alternative to the one developed by Cheng. In situations where the NC programs are using other software or manually, this system would be used for chatter detection. This approach also eliminates the need for any post-processing work that may need

to be done to generate NC programs from external tool path generation routines. Additional problems that may arise from setting up communication links with CAD packages are also avoided, since the NC code files have been generated externally.

It can thus be seen that depending on the type of application, the user has two systems to choose from to optimize his machining operation. Both systems offer the advantage that the user does not need to concern himself with chatter considerations when designing the part programs, since this entire process has been automated. In addition, both systems generate chatter free NC programs as their output, thus saving considerable machining time in verification cuts and possible remachining cuts to remove chatter marks.

Chatter Considerations and Limits of Stability

A brief overview of the literature involved in the process of chatter generation is provided in this section, in order to emphasize the importance of monitoring this phenomenon. Chatter, which is the occurrence of self-excited vibrations causes undulations in the finished surface and leaves behind what are known as chatter marks. These marks decrease the fatigue stiffness of the machined part and are removed by costly and time consuming manual finishing. Self-excited vibrations also amplify the nominal cutting force, and

when involved in the machining of hard materials, this can cause tool breakage or destruction of the workpiece. In order to avoid the above mentioned situations, a limiting depth of cut for chatter known as the limit of stability is established. This means that the metal removal rate is reduced for the whole cutting process because of the lowest allowable depth of cut, although the occurrence of chatter could be only for a short part of the tool path after which a larger limit of stability could be potentially used.

Regeneration of waviness is the mechanism responsible for chatter generation (Tlusty, 1985). Figure 1.1 shows the mechanism of regenerative chatter for a single degree of freedom system. The projection F cos(ß-∝) of the force excites the mode in the X direction and the projection X cos∝ of the vibration in the Y direction will modulate the chip. The conditions that can activate the regeneration of waviness mechanism and produce chatter depend on the relative transfer function between the machine and the workpiece, and the conditions of the cut (workpiece material K_s, spindle speed n, feed rate f, and chip width b). Figure 1.2 depicts the plotted transfer function in the complex plane for a frequency ratio f/f_n from zero to infinity. The oriented transfer function is given by:

$$G(w) = \frac{Y(w)}{F(w)} = \cos(ß-∝)\cos∝ \frac{X(w)}{F(w)} = \mu\, G(d) \qquad (1.1)$$

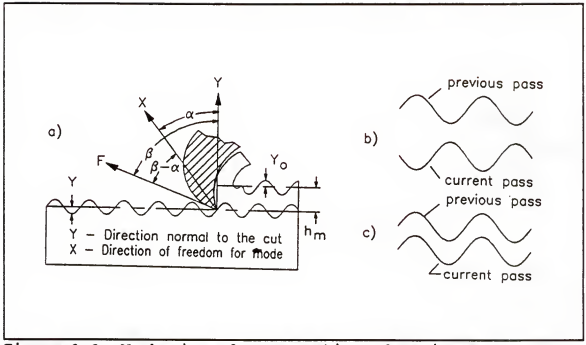

Figure 1.1 Mechanism of regeneration of waviness

where $\mu = \cos(\beta-\alpha)\cos\alpha$ is the directional factor, and G(d) represents the direct transfer function. Referring to Figure 1.1 the surface of the previous pass is given by,

$$y_o = Y_o \sin wt \tag{1.2}$$

and the current pass by

$$y = Y \sin(wt + 2\pi N + \epsilon) = Y \sin(wt + \epsilon) \tag{1.3}$$

where N is an integer and ϵ is the phase shift. The chip thickness is expressed as the sum of the mean chip thickness without vibration and the varying chip thickness between two passes.

$$h = h_m + h_v \tag{1.4}$$

which can be rewritten as

$$h = h_m + y_o - y \tag{1.5}$$

The force involved in the cut is expressed as

$$F = K_s bh = K_s b(h_m + h_v) \tag{1.6}$$

where K_s is the cutting stiffness of the material in N/mm^2 and b is the chip width. Figure 1.3 depicts the above process using the classical control system theory.

At the limit of stability any vibration would remain constant without decaying or increasing. Thus the absolute values of amplitudes for two subsequent vibrations are equal.

$$|Y| = |Y_o| \tag{1.7}$$

According to the Nyquist criterion, the limit happens when the open-loop transfer function of Figure 1.3 has the value -1.

$$K_s\ b_{lim}\ G(w)\ (1 - e^{-j\epsilon}) = -1 \tag{1.8}$$

or

$$b_{lim} = \frac{-1}{K_s\ G(w)\ (1 - e^{-j\epsilon})} \tag{1.9}$$

Since b_{lim} is always real and K_s is chosen to be positive real, the above equation can be justified only if $G\ (1 - e^{-j\epsilon})$ is real. This condition can be satisfied if

$$G\ (1 - e^{-j\epsilon}) = 2\ Re(G) \tag{1.10}$$

Thus

$$b_{lim} = \frac{-1}{2\ K_s\ Re\ (G(w))} \tag{1.11}$$

$$= \frac{-1}{2\ \mu\ K_s\ Re\ (G(d))} \tag{1.12}$$

For a system with two degrees of freedom in mutually perpendicular directions the equation for b_{lim} becomes

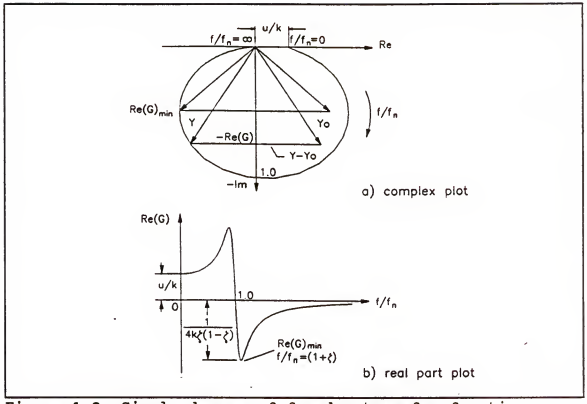

Figure 1.2 Single degree of freedom transfer function a) Complex plot b) Real part plot

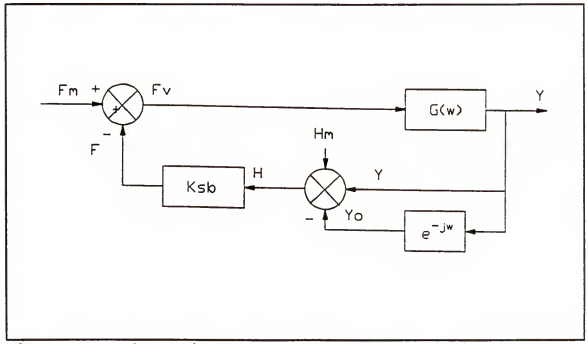

Figure 1.3 Block diagram of the chatter mechanism

$$b_{lim} = \frac{-1}{2\ K_s\ (\mu_x Re(G_x) + \mu_y Re(G_y))_{min}} \qquad (1.13)$$

where $Re(G_x)$ and $Re(G_y)$ are the real parts of the transfer function measured in the X (direction of feed motion) and Y (perpendicular to feed motion) directions respectively. The directional orientation factors μ_x and μ_y in Eqn. (1.13) take into account the dynamics of the machine structure and are dependent on the machining direction and the mode of milling (up or down).

Review of Literature

Object representation is one of the key processes in any design and production scheme. Drawings were adopted as the primary medium of geometric specification, with the 2-D wireframe models providing an early start in this direction. The internal representations in these models consisted of simple lists of lines and arcs. The 3-D wireframe systems developed in the 1970's were an extension of the above 2-D systems, where the 2-D lines and arcs were projected to obtain 3-D space curves, which were useful in producing multiple orthographic, isometric and perspective views. Wireframe models have some inherent deficiencies such as the ambiguous representation of solids and objects with curved surfaces (Requicha and Voelcker, 1982).

Remedies for the above mentioned shortcomings led to the development of a new generation of powerful systems known as the solid modeling systems. The term "solid modeling" encompasses an emerging body of theory, techniques and systems focused on informationally complete representations of solids - representations that permit any well defined geometrical property of any represented object to be calculated automatically (Requicha and Voelcker, 1983). The three commonly used representation schemes are the Decomposition models, Constructive models and Boundary models (Mantyla, 1988). Decomposition models describe solids through a combination of some basic building blocks. Exhaustive Enumeration techniques represent the solid as a list of tiny cubes contained in the solid, and form a regular subdivision of the space. More efficient adaptive space subdivision schemes include the Octree representation, which uses a recursive subdivision of the space of interest into eight octants that are arranged into an 8-ary tree.

Half-spaces form the basic modeling primitives for half-space models, and are combined by Boolean set operations like union, intersection and set difference. However, combinations of half-spaces may result in unbounded point sets, which led to the development of the Constructive Solid Geometry (CSG) approach which uses only bounded point sets as its primitives. A CSG tree is usually made up of the solid primitives along with associated set operations (union, intersection and

difference) and rigid motions (translation or rotation). An example of its use is in the modeling of the subassemblies of a design, which after appropriate transformations are used several times in the CSG tree representing the assembled design.

Boundary models (B-rep) represent a solid indirectly through a representation of its bounding surface. The surface of the object is divided into an enclosing set of faces, each of which is represented in terms of its bounding polygon, which in turn is represented in terms of edges and vertices. In addition to geometric information such as face and curve equations and vertex coordinates, a boundary model must also represent how the faces, edges and vertices are related to each other. All information of the geometry of the entities is bundled under the term "geometry" of a boundary model, and similarly information of their interconnections under the term "topology."

The geometric simulation of NC machining processes is a very important step towards the development of efficient verification systems. Simulation implies that a model of the current state of the machined object is always present. This feature makes it just as easy to find areas of unremoved material as it is to find interferences between the cutting tool and workpiece. Some of the packages which have been developed to carry out these simulations include the PADL system developed by Voelcker and Hunt (1981), and the TIPS

solid modeling package by Fridshal et al. (1982) at General Dynamics. The solid modeling approach, however, is computationally expensive and entails the use of a large amount of system memory.

A different approach for simulating three-axis machining was invented by Anderson (1978). He divides the base of the object into squares, and keeps track of the cut height above each square. Each square starts with the value of the height of the stock. Each tool movement updates the heights of the squares it passes over if it cuts lower than the currently stored height. The square grid makes it easy to determine which squares lie under the tool path and which can be ignored.

The "point-vector" technique of Chappel (1983) approximates the surfaces of the part by selecting a set of points lying on the surface. Direction vectors are created normal to the surface at each point. A vector extends until it reaches the boundary of the original stock or intersects with another surface of the part. The simulation of the cutting caused by a tool movement can be obtained by finding the intersection of each vector with that tool movement's envelope. The length of a vector is reduced if it intersects the envelope. Oliver and Goodman (1985) have improved on Chappel's technique by using a graphical image to select the points defining the surface.

Wang (1986) developed the extended Z-buffer algorithm, where both the front and back Z values of a volume are stored. The swept volume of a tool movement is replaced by a polyhedral approximation. A scan line plane intersected with this polyhedron determines the Z depth of each pixel associated with the graphic image of the swept volume. The workpiece Z buffer is then modified by comparing it with the swept volume Z buffer. Upon completion of the simulation, the Z buffer of the workpiece and that of the desired mathematically defined surface can be compared to find unacceptable differences.

Van Hook (1986) developed an alternate extended Z-buffer scheme where he precomputes a pixel image of the cutting tool and performs Boolean subtractions of the cutter from the workpiece as he steps along a tool path. This limits his approach to three axis machining where the orientation of the cutting tool does not change.

These above mentioned point vector techniques have several advantages over the solid modeling approach. Their run time grows linearly with the number of tool movements, so they can handle sculptured surfaces with several thousand tool movements. They also provide an easy way to measure exactly the difference between the cut surface and the desired surface. One drawback is that the approximation of surface by a set of points introduces errors. This is not a problem as long as the size of the error introduced is small enough for

a given application. Jerard et al. (1986) have shown that by a proper choice of surface points, this simulation error can be bounded and made as small as possible. A vector whose direction is parallel to the longitudinal axis of the cutting tool is passed through each of these points. As the simulation progresses, the intersection points of the vectors with the cutting tool are calculated and can be examined to see if they are within a specified tolerance of the desired surface.

With the development of efficient simulation techniques, it has been possible to suitably monitor the machining operations. Takata et al. (1989) have developed a cutting simulation system for machinability evaluation. The system consists of a geometric simulation as well as a physical simulation. The geometric simulation utilizes a combination of the extended Z-buffer technique (Hook, 1986) and the swept volume generation technique (Wang, 1986) for the representation of the workpiece and the cutting operation. In the physical simulation, first the cutting force vector at a certain instance is calculated by using a force model. The tool deflection induced by this force vector is then calculated by a static tool deflection model. The machining error is estimated by transferring the tool deflection into the workpiece geometry.

Sungurtekin and Voelcker (1986) have studied the graphical simulation and automatic verification of NC machining programs. The simulator seeks to display the effects

of the cutter motions on the workpiece. They enable the NC programmers to detect visually potential collisions and gross programming errors. The verifier on the other hand seeks to determine automatically whether an NC program will produce a specified part without undesirable side effects like collisions with fixtures, cutter breakage, etc. Simulators and verifiers both require solid modeling facilities, but simulators need only fast display generators for Boolean combinations whereas verifiers require more varied and powerful facilities.

It is very obvious from the above discussion that there is a very urgent need for the development of verification systems that monitor the cutter engagements with the workpiece. The various simulators that have been developed could be adequately used to model the machining process, thus leaving the programmer with the unfinished task of developing new algorithms to extract the various process parameters (axial and radial depths of cut). Altintas and Yellowley (1987) researched some strategies for the real time identification of current geometry of cut, so that analytical estimates of expected forces and other parameters may be made. Their algorithm allows the independent identification of both axial depth and radial width of cut based only on two orthogonal force measurements in the plane perpendicular to the machine tool spindle. However, since the proposed verifiers are off-line systems, it is necessary to extract

these parameters from the computer models preferably during the simulation process.

Melkote and Kendall (1989) have worked on the extraction of the cutter engagements from the NC tool trajectories. Their work has proved to be very promising so far. The workpiece geometry is represented by a 3-D B-rep model and a current version of it is always maintained in the database. Intersections that provide engagement dimensions are computed using this current version and the surfaces defined by the tool trajectory that are obtained from the parsing and interpolation of the NC instruction set. Their system is being currently used to preprocess the NC program to generate a run time control file for an intelligent supervisory control system. This run time program could be used to collect appropriate sensor data and in a postprocessor mode reprogram the NC tape.

The above system developed by Melkote and Kendall provides the basic foundation for the verification system for stability in milling, which is the topic of the current research. It has been pointed out repeatedly that solid modeling systems are expensive and have severe memory constraints. Hence, the current system has been geared towards application on a two-dimensional wire frame model. The entire system can be broken down into two stages, namely, the simulation stage and the monitoring stage. This modular construction lends a high degree of flexibility to the effect

that this system could be easily modified to accommodate 3-D models of the workpiece. This would entail the replacement of the current models by a 3-D wireframe model or solid model.

Outline of the Present Research

The primary objective of the current research was to develop a verification system that could preprocess a NC program and automatically detect regions of instability in the cutting paths, where the phenomenon of chatter will occur. The optimization process would involve a reprogramming of the NC code automatically with new values for the axial depths of cut, which are determined from a pre-programmed database.

This system can be run on any PC with graphics capacities. The programming language used is C, since it facilitates very easy representation of complex data structures and also provides dynamic allocations of memory. A graphical display of the tool path is also provided, which serves as an effective simulation of the machining process. Color codes are used to distinguish between regions of stable and unstable machining. A new NC program file is also generated, which contains the reprogrammed tool paths with the stable values of the axial depths of cut.

Chapter 2 of this dissertation is devoted to a detailed description of the various inputs to the verification system. The schemes used to represent the geometry and preprocess the

NC code file are described. Chapter 3 concentrates on the various Boolean schemes which are necessary to calculate the intersection between the tool swept area and the current model of the workpiece. The techniques used to update the current model of the workpiece and also generate the tool removed area are outlined.

Chapter 4 contains the algorithms for extracting the axial and radial immersions from the intersection points calculated above. Schemes to determine the directions of feed and the mode of milling (up or down) are also covered in this chapter. A detailed description of the database which contains the limit values of stability, and the final optimization process which involves the NC reprogramming is provided.

The application of the verification system on a spiral-in pocketing routine is demonstrated in Chapter 5. Some special changes that need to be made for extensive application of this system in other pocketing routines are outlined. A graphical output of the simulation and a listing of the reprogrammed NC file is also provided. New schemes which will allow the system to be extended to spiral-out and zig-zag pockets are covered in Chapter 6. Inclusion of islands in the profiles are also dealt with. Chapter 7 contains a detailed analysis of the various pros and cons of the developed system along with some suggestions for future research that could exploit the system's full potential.

CHAPTER 2
WORKPIECE AND TOOL SWEPT AREA REPRESENTATIONS

The literature review presented in Chapter 1 discussed the various representation schemes used to model the workpiece and the tool motion. Most of these schemes were predominantly geared towards the solid modeling approach. An analysis of the various decisive factors involved in the selection of a representation scheme to be used in the verification system, along with a detailed description is provided in this chapter.

Wireframe Models

The simplest form of object representation is the wireframe system, where the basic entities are lists of lines and arcs. They have some inherent advantages in that, large collections of semiautomatic application codes are available for wireframe systems, many thousands of parts have been defined through wireframe systems, and a new generation of PC-based systems makes wireframe technology accessible to small firms. The representation scheme chosen for use in the present research was a two dimensional wireframe model.

The decision process involved in the wireframe vs solid model selection should take into account the complexity of the objects being dealt with and the memory constraints involved. This verification system was primarily envisioned for initial applications involving simple geometries such as rectangular pockets with or without islands. The main focus here was to come up with an effective system for the off-line detection of chatter conditions without being mired too deeply in the problems of object representation. Commercial verification systems which have been developed in the past have concentrated their efforts on features such as tool and fixture collisions, gouging, and detection of areas of unmachined material. Although a lot of work has been done in the area of on-line chatter detection, there is as yet no complete off-line detection system. The research presented in this dissertation has made the first steps towards the development of such a system.

As mentioned above, the geometries considered here consist of basic entities such as lines and arcs. It is reasonable to assume that a majority of the machining applications would encompass geometries of the above variety. These kind of geometries lend themselves very easily to the wireframe mode of representation. In addition, the database of limit values described later in Chapter 4, which is used for monitoring purposes stores information at present for two and a half axis machining cases. Thus, the 2-D models were chosen

over the 3-D ones. Under this scheme cutting planes are used to represent the axial depths of cut. The system in its present format can be used to handle only discrete increases in axial depths and not continuous changes. By updating the cutting plane with each change in the axial depth of cut, the changes in the third dimension (Z- direction) are tracked. Finally, the system was also developed primarily for PC based applications initially, and this introduced the memory constraints for data storage as another factor in the selection process.

Workpiece Representation

A preliminary representation scheme for the workpiece geometry in the verification system is presented below. The shortcomings of this approach along with the various implemented changes are presented in Chapter 5. The C programming language provides features which support the desired wireframe mode of representation. The entire workpiece can be visualized as a collection of its edges. The edges which make up the workpiece can be classified into three basic entity groups namely, line, clockwise arc and counter-clockwise arc. Each of these entities has some distinguishing characteristics which describe it in its entirety. Figure 2.1 shows a detailed description of the various entities and their corresponding characteristics. Early programming languages

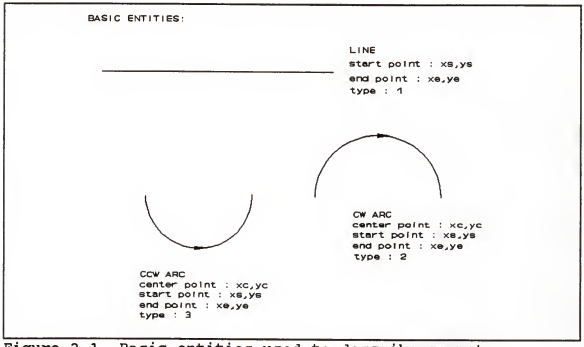

Figure 2.1 Basic entities used to describe geometry

such as FORTRAN offered features like arrays to represent data in a sequential fashion. However, a C specific feature called a structure allows large and cumbersome data to be stored in a very compact manner. These structures hold the pointers to addresses in memory where the data is stored. The structure used to depict the three basic entities is shown below:

```
struct edgedefine {
                    int element;
                    double points[6]; }
```

where edgedefine is the user supplied name which can be used later in the program to define more variables of this structure type. The structure itself has two components, the integer variable "element" and the double precision floating point array called "points" with a maximum size of 6. The

variable "element" is the index which is used to distinguish between the type of entity being defined. In the present scheme, element is assigned a value of 1 for lines, 2 for clockwise arcs and 3 for counterclockwise arcs. The array "points" stores the information about a particular entity. The line entity has four distinct characteristics, the x and y co-ordinates of the start and end points. The arcs have, in addition to the above entries, the x and y co-ordinates of the center point. Thus each edge of the workpiece is represented by an individual structure of type edgedefine.

The entire workpiece, which is regarded as a collection of the individual edges, can be represented by linking the various structures which define the edges. A linked list establishes a connection between the individual edges and thus defines the entire workpiece. It assigns two pointers for every edge called "prev" and "next", which store the addresses of the previous and next edges with respect to the current edge. This is referred to as a doubly linked list, since it is possible to traverse around the list both in the forward and backward direction. This list can be constructed as a closed or circular list by setting the next pointer for the last edge to the first edge, and the previous pointer for the first edge to the last edge in the list. Figure 2.2 shows a circular doubly linked list representation of a square workpiece with four edges. The first edge in the list is referred to as "start" for purposes of clarity. The structure edgedefine will

now have to be modified to take into account the effects of the two pointers, prev and next and is defined as follows:

```
struct edgedefine {
                    int element;
                    double points[6];
                    struct edgedefine *prev;
                    struct edgedefine *next;  }
```

It is assumed that a previous knowledge of the starting workpiece geometry is available. An external data file would be the simplest way of storing this information. The data file

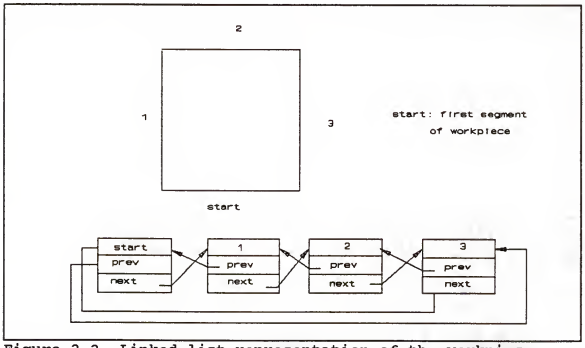

Figure 2.2 Linked list representation of the workpiece geometry

is then scanned line by line, and the information is read into structures of type edgedefine. It is assumed that the edges are arranged in a clockwise orientation from the starting

edge. Thus a line with start point (2.0, 1.0) and end point (3.0, 3.0) would be represented in the data file as follows:

```
1  2.0  1.0  3.0  3.0
```

where the first entry "1" depicts the entity type, which is a line in this case. A clockwise arc from (0.0, 0.0) and ending at (2.0, 0.0) with center at (1.0, 0.0) is represented as:

```
2  1.0  0.0  0.0  0.0  2.0  0.0
```

The second and third entries represent the co-ordinates of the center point. The entire data file is scanned until all the edges have been constructed and the final linked list representation of the workpiece has been set up. The pointer "start" which points to the first edge in the list, is used to store the starting address of the entire list.

Tool Swept Area Representation

Most of the present day controllers of CNC machines take as input the instruction set either in the punched tape format or as a series of tool movements encoded in a computer data file. Neither of these input forms contain explicit feature information. Hence it is necessary for the CNC instruction set to be parsed in order to create a feature based description of the cutting operations. The NC code file is preprocessed by scanning one block of code at a time. A parser routine sifts out the X, Y and Z values from every block of code and ignores the remaining entries. The X, Y, Z co-ordinates of the current

position of the tool and it's destination position can be calculated from successive scans. A few blocks of NC code are shown below:

```
N10 G01 G90$
N20 M03 S600$
N30 Z-0.5 F50.0$
N40 X+0.5 Y+0.5 F10.0$
N50 Y+1.5
```

When block #50 is scanned the current values of the tool position are, X = 0.5, Y = 0.5, Z = -0.5, and the tool destination values are X = 0.5, Y = 1.5, Z = -0.5. These values are updated with the scanning of every block of code.

Once the current and destination positions have been calculated, the tool trajectory and swept area can be easily represented. A representation of the tool swept area is necessary, since it's intersection with the workpiece geometry needs to be determined for further calculations. The tool swept areas are also represented as 2-D models, with the doubly linked circular list forming the basis of their construction. This is in contrast to the representation used by Melkote and Kendall where the tool motion is depicted by a volume in 3-D space. This volume in turn is approximated by three planar faces, one on either side of the tool and one at the bottom.

Figure 2.3 shows the linked list depicting a single tool swept area for the 2-D model. The swept area for each linear

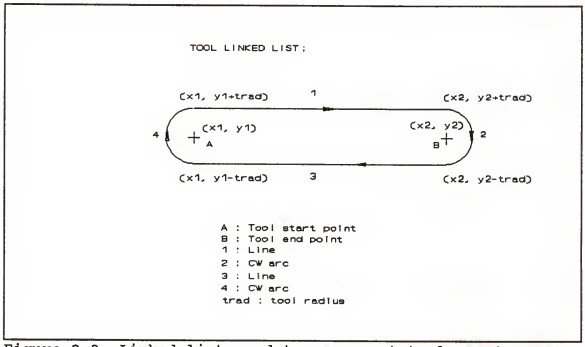

Figure 2.3 Linked list used to represent tool swept area

movement of the tool can be represented as a slot (Cheng, 1991). This slot can be split up into two lines and two half circles. With a knowledge of the tool diameter in addition to the current and destination positions of the tool, the geometric data that define these four entities can be easily calculated. In Figure 2.3 (x1, y1) depicts the current position of the tool, and (x2, y2) the destination position. That figure illustrates a tool motion in the horizontal direction (X direction). Movements in other directions such as (X+Y, X-Y) are also taken into account by introducing another variable, which is the slope of the tool trajectory. This is the line that is drawn between the points (x1, y1) and (x2, y2). Figure 2.4 shows the tool swept area representation in the X+Y direction.

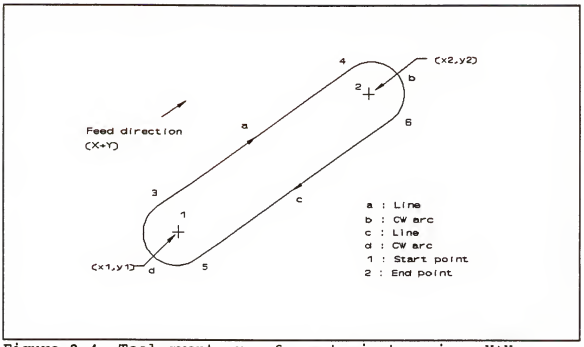

Figure 2.4 Tool swept area for a trajectory in a X+Y direction

The slope ß of the tool motion line in this case is given by,

$$ß = \text{acos}\ (\ (x2 - x1)\ /\ d\ ) \tag{2.1}$$

where,

$$d = \text{sqrt}\ (\ (x2 - x1)^2 + (y2 - y1)^2\ ) \tag{2.2}$$

If y2 < y1 then,

$$ß = (2\pi - ß) \tag{2.3}$$

The x and y coordinates of the points 3,4,5 and 6 in Figure 2.4 are given below.

$$x3 = (\ x1 + r * \cos(ß + \pi/2)\ ) \tag{2.4}$$

$$y3 = (\ y1 + r * \sin(ß + \pi/2)\ ) \tag{2.5}$$

$$x4 = (\ x2 + r * \cos(ß + \pi/2)\ ) \tag{2.6}$$

$$y4 = (\ y2 + r * \sin(ß + \pi/2)\ ) \tag{2.7}$$

$$x5 = (\ x1 + r * \cos(ß - \pi/2)\ ) \tag{2.8}$$

$$y5 = (y1 + r * \sin(\beta - \pi/2)) \quad (2.9)$$

$$x6 = (x2 + r * \cos(\beta - \pi/2)) \quad (2.10)$$

$$y6 = (y2 + r * \sin(\beta - \pi/2)) \quad (2.11)$$

The entity r used in the above equations is the tool radius. The tool swept area representation however must be modified in the case of circular interpolation. The two line segments used in the representations for straight line motions are now replaced by two arcs. Figure 2.5 shows the swept area representation for such a case. The geometry is now represented by four arcs with (x1,y1) being the start point and (x2,y2) the destination point. Additional information in the form of two vectors I and J is obtained from the NC instruction set for such a motion. These two vectors help to determine the point (xc,yc), which is the center point for the arcs 1 and 3. The following equations are used to determine the coordinates of the various points p, q, r and s shown in Figure 2.5.

$$xc = (x1 - I) \quad (2.12)$$

$$yc = (y1 + J) \quad (2.13)$$

The unit vectors i_1 and j_1 are given by

$$i_1 = (x2 - xc) / d \quad (2.14)$$

$$j_1 = (y2 - yc) / d \quad (2.15)$$

where $$d = \text{sqrt} ((x2 - xc)^2 + (y2 - yc)^2) \quad (2.16)$$

$$xp = x2 - (i_1 * r) \quad (2.17)$$

$$yp = y2 - (j_1 * r) \quad (2.18)$$

$$xq = x2 + (i_1 * r) \quad (2.19)$$

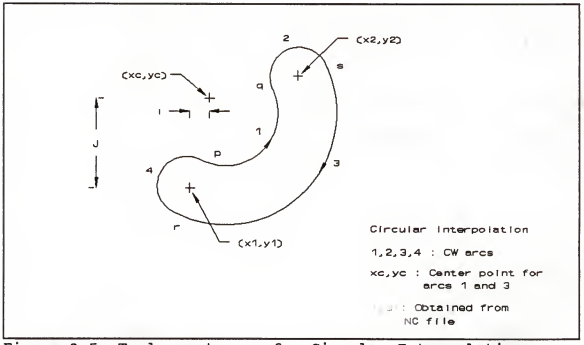

Figure 2.5 Tool swept area for Circular Interpolation

$$yq = y2 + (j_1 * r) \tag{2.20}$$

The unit vectors i_2 and j_2 are calculated next.

$$i_2 = (x1 - xc) / d1 \tag{2.21}$$

$$j_2 = (y1 - yc) / d1 \tag{2.22}$$

where $d1 = sqrt\ (\ (x1 - xc)^2 + (y1 - yc)^2\ )$ (2.23)

$$xr = x1 - (i_2 * r) \tag{2.24}$$

$$yr = y1 - (j_2 * r) \tag{2.25}$$

$$xs = x1 + (i_2 * r) \tag{2.26}$$

$$ys = y1 + (j_2 * r) \tag{2.27}$$

A special case arises when the radius of arc 3 is very small. Figure 2.6 shows such a case where the geometry contains only three arcs. The calculations are performed in a similar manner as shown above for this special case also.

The entities that make up the tool linked list are also represented by structures similar to the ones used for the

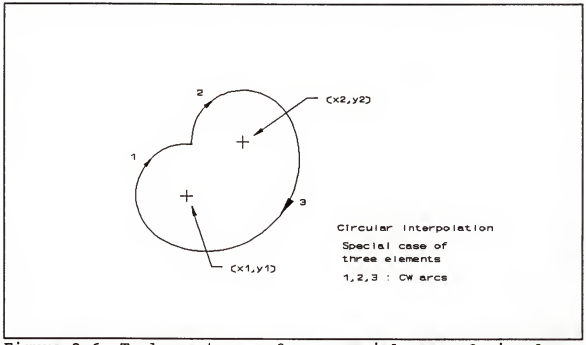

Figure 2.6 Tool swept area for a special case of circular interpolation (3 elements)

workpiece representation. Each of the entities are assigned the prev and next pointers, thereby providing for sequential traversal through the list. With the completion of these representations, the next step involves the implementation of algorithms to determine the intersection points between the workpiece and tool swept area. These techniques are highlighted in greater detail in the next chapter.

CHAPTER 3
BOOLEAN OPERATIONS

In geometric modeling, simple shapes often referred to as primitives are combined to form more complex shapes. Boolean operations such as set intersection, union and difference are used to perform the above operations. The algorithms that perform these operations must produce as output, objects that are closed set of points having boundary and interior subsets and preserve the dimensionality of the initial objects.

Outline of Algorithms

Mortenson (1985) has developed several algorithms to carry out the various Boolean operations like intersection and difference. Figure 3.1 shows the results of some of these operations on two simple polygons A and B. The arrows indicate the direction of parametrization. Note that both A and B are parametrized in a consistent direction. In this case, they are both counterclockwise. By maintaining this consistency in orienting the polygon's edges, a convention can be established to define which side contains points inside the polygon. The points 1, 2, 3, and 4 in Figure 3.1 denote the points of intersection between the edges of A and B. A detailed

description of the intersection algorithm to detect the intersections between the workpiece model and the tool swept area is provided in the following sections.

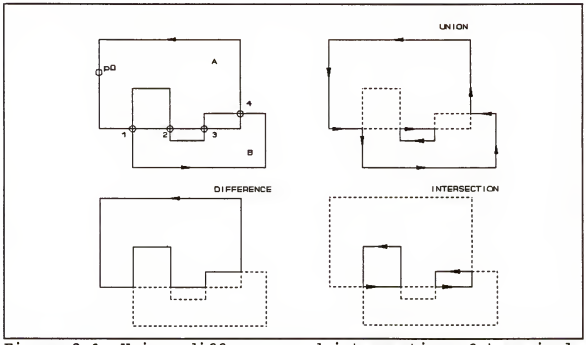

Figure 3.1 Union, difference and intersection of two simple polygons

Determination of Intersection Points

In order to effectively represent the machining operation, it is necessary to store a current model of the workpiece at all times. With the execution of each block of code in the NC instruction set, this model needs to be updated to account for the removal of stock. The first step in this process involves the determination of all legal intersection points between the workpiece model and the tool swept area.

The term legal is used in this context because it is common to have several points of intersection, but only a few of them actually play a significant role in the updating process. The algorithms need to be capable of discarding these redundant points.

Figure 3.2 shows an arbitrary workpiece model A, and a representation of a tool swept area in the -X direction, B.

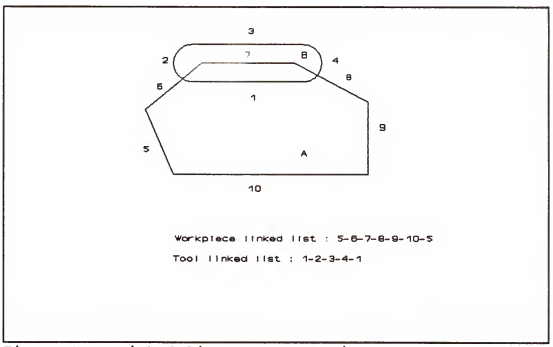

Figure 3.2 Linked list representation of A and B

The circularly linked list representation of A consists of the edges 5-6-7-8-9-10-5. The tool swept area is denoted by the edges 1-2-3-4-1. A discussion of these representations was presented in Chapter 2. All intersection points between A and B are determined in a sequential fashion by scanning the linked lists of both A and B and then selecting two edges to check for possible intersections. In Figure 3.2 the first two

edges selected would be the starting edges 5 and 1. This is followed by the edges 6 and 1, 7 and 1 and so on till all the edges in the linked list of A are exhausted. The above procedure is repeated with the other edges (2, 3 and 4) of B and all possible intersection points are determined. To simplify the above checking procedure certain assumptions are made to discard invalid intersection points. For example, no intersections are said to exist when two lines are parallel or coincident, or when two arcs have the same center. Figure 3.3 shows a case where it is possible to have four points of intersection. The points p and q are not legal (useful) points for this particular case and need to be discarded. A check for such a case needs to be included to prevent the inclusion of these points in the updating algorithm. It is envisioned that the number of these checks will increase as the complexity of the geometries involved increases. The pseudo-code for the above operation is provided below:

```
tmp = start;  / pointer to edge 5 for workpiece /
tmp1 = start1; / pointer to edge 1 for tool /
do
  {    / do loop for updating tool edges /
     do
         {  / do loop for updating workpiece edges /
           intersect (tmp, tmp1); /intersection function/
           tmp = tmp -> next;  /update tmp to next edge/
          }  while (tmp is not equal to start);
```

```
        tmp1 = tmp1 -> next;  /update tmp1 to next edge/
    }  while (tmp1 is not equal to start1);
```

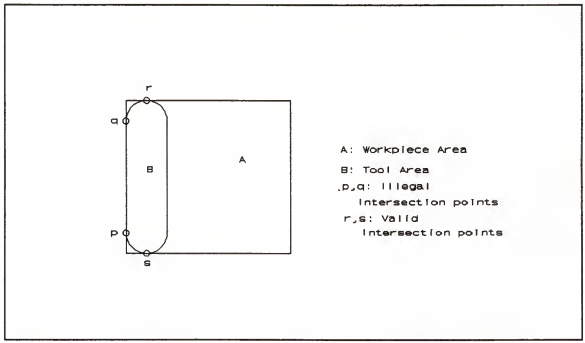

Figure 3.3 Points of intersection determined during a Slotting Pass in the Y direction

For this example, it is possible to have only two intersection points at most for a given tool swept area. If the number of points is one or zero, there is a possibility of two cases: (i) A is inside B, or (ii) A and B have no area in common at all. Both cases can be distinguished by first choosing any point on the boundary of A. If this point is inside B, it implies the existence of case (i) which means the current tool movement is the last one, for there is no area left to be cut. If this point is not inside B, it can be deduced that there is no common area indicating a tool traversal in air. It should be noted here that difficulties in intersection detection

could arise if the tool trajectory was completely inside A and did not cross any of its boundaries. Such a situation arises in the case of the spiral-out pocketing routine. These situations are dealt with in more detail in Chapter 6.

Difference and Intersection Operations

After all valid intersection points have been determined, the next step involves the manipulation and linking of selected edges from A and B to obtain updated workpiece models as well as the area of the removed material for each tool motion. Both A and B are split into two pairs of half loops at the intersection points. In Figure 3.4 p and q denote the two intersection points between A and B. The case depicted here is a tool motion in the X direction for a down milling cut. The two edges in the linked list of A that need to be split are 6 and 8. The corresponding edges for B are 2 and 4. The edge 6 is split into two smaller edges 6' and 6" with the split being made at the point q, while edge 8 is broken into 8' and 8" at the point p. It is now possible to break the circular linked list of A into two open half loops. The first half loop of A (loop1) is represented by 8"-9-10-5-6'. The second half loop (loop2) is comprised of the edges 6"-7-8'. The above procedure is repeated for the tool edges 2 and 4 to obtain the half loops 2"-3-4' (loop3) and 4"-1-2' (loop4). Figure 3.4 also depicts the linked list representations of these half loops.

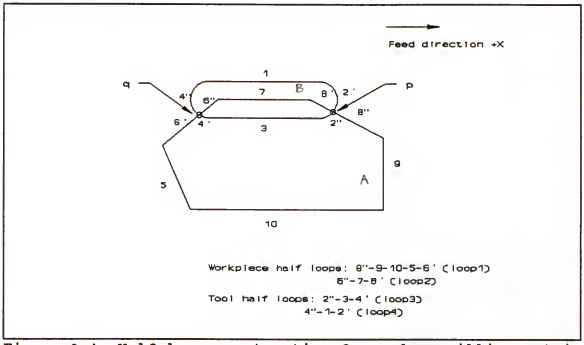

Figure 3.4 Half loop construction for a down milling cut in X direction

The Boolean operations such as intersection and difference now involve manipulation of the above half loops to obtain the desired final representation. For example, the difference operation A - B involves the determination of a half loop of A that is outside B. First a point p_0 on the starting segment of loop1 is selected and it is determined whether or not it lies inside the area B. This can be performed by using the concept of the winding number. The winding number of a closed loop is defined as follows:

$$W=\sum_{i=1}^{n} W_i \qquad 3.1$$

where n is the number of edges in the closed loop.

$$W_i = \frac{1}{2\pi} \int \frac{(x_i(t) - p_x)\, y_i'(t) - (y_i(t) - p_y)\, x_i'(t)}{(x_i(t) - p_x)^2 + (y_i(t) - p_y)^2}\, dt \qquad 3.2$$

where (p_x, p_y) represents the x and y co-ordinates of point p_0, and $(x_i(t), y_i(t))$ denotes the co-ordinates of any point along the i'th edge of the loop. $x_i'(t)$ and $y_i'(t)$ are the first derivatives of $x_i(t)$ and $y_i(t)$ with respect to t. The exact solution of W_i can be obtained by solving the integral if the edge involved is a line or arc. Other numerical techniques may have to be applied for more complicated curves.

The winding number reveals the relationship between a closed loop and any arbitrary point p_0. Three possible cases are shown in Figure 3.5. If a point p_0 lies inside a closed counterclockwise loop, the winding number equals 1. The winding number takes a value of -1 for an interior point in a clockwise loop and 0 when the point lies outside the loop. Other applications of the winding number include classifying the direction of a closed loop. This could prove to be useful when complicated geometries with several islands are encountered, since the validity of the various half loops can be checked by looking at their orientation.

The winding number of any point on the starting edge of "loop1" with respect to the area B is calculated using the above equations. If this value is zero, then this is the desired half loop that lies outside B. Any other value will indicate the selection of "loop2" as the desired half loop. This procedure is repeated for the tool swept area half loops

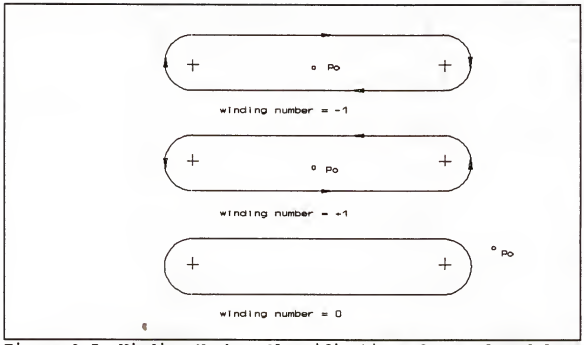

Figure 3.5 Winding Number Classifications for a closed loop

to check whether any point on the starting edge of the loops lies inside A. Since the workpiece orientation was chosen as clockwise, the desired winding number value should be -1. For the example case in Figure 3.4 the two selected loops are "loop1" and "loop3."

The case considered above was a case of down-milling. Figure 3.6 shows the half loop construction and linking for an up-milling case for a tool motion in the X direction. As before, p and q are the two points of intersection. The linked lists depicting the various half loops are shown in the figure. The winding number calculations are performed in a similar manner as mentioned above. The two half loops selected in this case are "loop2" and "loop4." The distinction between the up-milling and down-milling cases can be seen from the

fact that the material to be removed is to the left of the line along the feed direction for the former and to the right for the latter case.

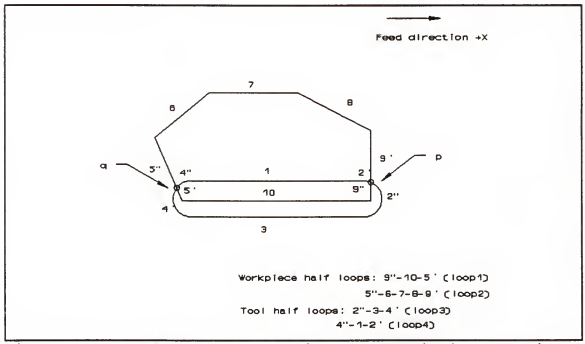

Figure 3.6 Half loop construction for up milling case in X direction

The difference operation will provide an updated model of the workpiece. It is this updated model which will be used for Boolean operation calculations on the next tool swept area. In order to maintain the clockwise orientation of the workpiece model it is necessary to reverse the direction of "loop3" in Figure 3.4 and "loop4" in Figure 3.6. The loops "loop1" and "loop3" (after reversal) are now combined together to form a new linked list which represents the updated workpiece model for the down-milling case. Figure 3.7 shows the linked list representation of the above difference operation. For the up

milling case, "loop2" and the reversed "loop4" are linked together to form the updated model.

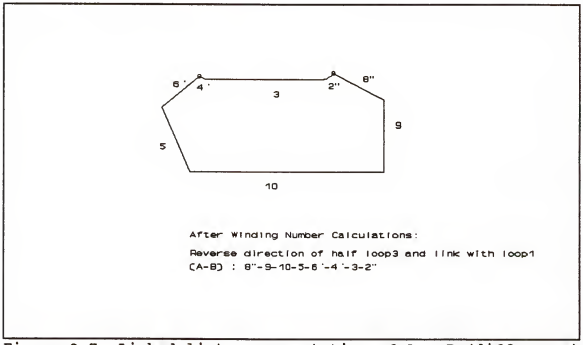

Figure 3.7 Linked list representation of A - B (difference)

The intersection operation A ∩ B provides the area of removed material for the current tool motion. The removed area is analogous to the swept volume determined by Melkote and Kendall in their system. This area is useful for the extraction of the cutter engagements which is covered in greater detail in Chapter 4. The removed area can be obtained by combining "loop2" and "loop3" (before reversal) to obtain the new linked list for the down-milling cut. Figure 3.8 illustrates the result of the above operation. For the up-milling case, the two loops which are linked are "loop1" and "loop4."

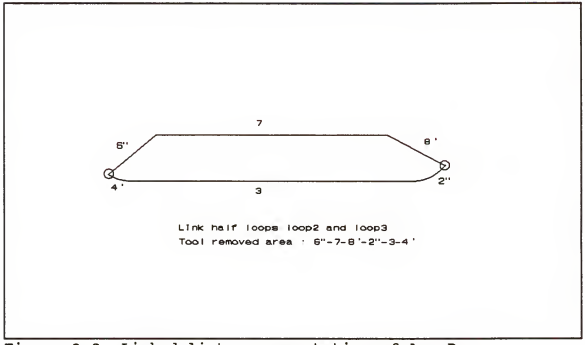

Figure 3.8 Linked list representation of A ∩ B (intersection)

If representations of the initial objects A and B are no longer necessary, they can be discarded and deleted from the modeling database. In any event, the resultant object is represented by reparametrized concatenations of the appropriate edges of A and B. The algorithms outlined above require several supporting algorithms. The principal ones are: a procedure for finding the intersection points; a procedure for determining if a point is inside or outside a closed polygon; and procedures tracing edges, forming loops and reparametrizing the loops. Since the representations used in the above schemes are restricted to lines and arcs, the intersection detection algorithm needs to check only for intersections between two lines, a line and an arc, and between two arcs. Other algorithms to check whether these

points lie on the appropriate arcs and lines are also necessary. The list of these supporting routines will probably grow with the inclusion of more complicated geometries.

It can be seen from the above discussion that the Boolean operations are very useful for representing the machining process. This process begins with a simple stock shape and then proceeds to remove material with various cutting tools until some final designed shape is achieved. A parallelism has been established between this subtractive machining operation and the difference operator. The entire process can thus represent the successive passes of the cutter of a machine tool by a sequence of difference operations on the initial model of the part's raw stock shape. The intersection operator is essential for determining the removed area which is necessary for further verification calculations.

CHAPTER 4
VERIFICATION TECHNIQUES

This chapter highlights the techniques used to extract the cutting parameters from the removed area and some verification algorithms that are used to predict chatter during the simulation process. This entire process can be separated into three distinct stages. Stage 1 involves the extraction of the cutter engagements (axial and radial depths) from the removed area, along with the determination of the mode of milling (up, down or slotting) and the direction of feed. Stage 2 is the comparison of the extracted axial and radial depths for the present tool position with the limits of stability stored in the previously developed database. It is determined at this stage whether the current tool motion will result in an unstable cut or not. Stage 3 involves the reprogramming of the NC file. A scheme is used to repeat the tool motions at lower axial depths in the event of chatter detection. Alternate correction techniques are also investigated for future applications such as the presence of internal corners. These three stages are described in detail below.

Extraction of Cutting Parameters

The cutting parameters of interest are the radial immersion, axial immersion, mode of milling and the direction of feed. As mentioned earlier, the removed area is necessary to aid in the calculations of the radial immersion. The representation of the machining process described in Chapter 3 is divided into very small positional increments (dx, dy). It is necessary to determine the radial and axial immersions at each of these instances. The radial immersions can be calculated by finding the intersection points between the tool representation and the removed area at each increment. This permits the determination of the start and end angles of the cut. Figure 4.1 shows a tool motion in the +X direction at half radial immersion. An arbitrary removed area is chosen and p1 and q1 represent the intersection points of the tool with the removed area at instant 1. The points p2 and q2 are the intersection points at instant 2 and so on.

It should be mentioned here that Chapter 2 described a scheme to represent the entire tool swept area from the start point to the destination point and not the cutting tool itself at each instant of the simulation. It is therefore necessary to develop a separate representation for the cutting tool at these separate instants. The C structure representing the tool is as follows:

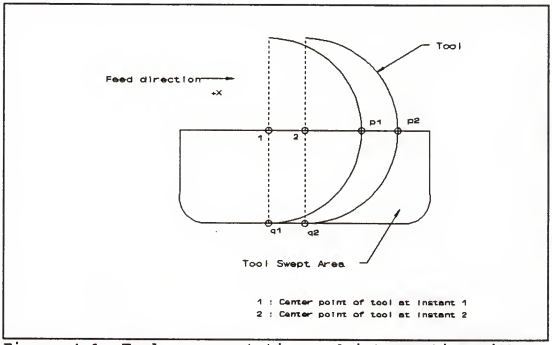

Figure 4.1 Tool representation and intersection with swept area

```
struct tool {
                int element;
                double points[6];
            }
```

where "element" has a value assigned 2 for a clockwise arc, since the cutting tool rotation is fixed in this orientation. The array "points" contains the x and y co-ordinates of the center, start and end points of the arc. Figure 4.2 shows the above representation used to depict the cutting tool. The center point for the arc can be determined from the dx and dy increments, while the start and end points can be calculated with a knowledge of the tool radius and the direction of feed. A sample calculation for a feed in the +X direction is shown in Figure 4.2. The tool radius can either be obtained from the

NC file (represented as a separate T code) or as an input from the user. The direction of feed can be calculated from a knowledge of the start and destination points of the current tool trajectory. For example let (xprev, yprev) represent the start point and (xnew, ynew) the destination point. If (xnew > xprev) and (ynew = yprev) then the direction of feed is +X. If (xnew < xprev) and (ynew = yprev), the direction of feed is -X. The other directions of feed are +Y, -Y, +X+Y, +X-Y, -X+Y, and -X-Y which can be determined by similar checks as shown above. Figure 4.2 shows the tool representations for only four directions of feed. Representations for tool motions in combined X and Y directions are similar, except that the slope of the tool trajectory line should be incorporated in the calculations of the start and destination points. Figure 4.3 shows the calculations used to represent the cutting tool for a feed in the X+Y direction. It should also be pointed out here that there is no necessity to use a linked list representation of the cutting tool since it is a single entity.

The intersection points between the cutting tool representation and the removed area are determined by using the same techniques as described in Chapter 3. The intersection detection algorithms used earlier can be applied here also. During steady state cutting conditions, the number of intersection points is usually two. The starting and ending angles of cut can then be calculated from these points.

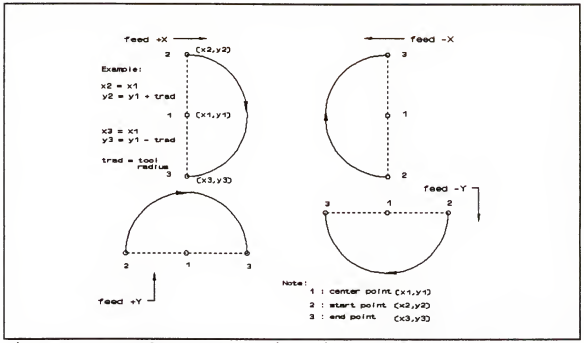

Figure 4.2 Tool representations in +X, -X, +Y, -Y directions

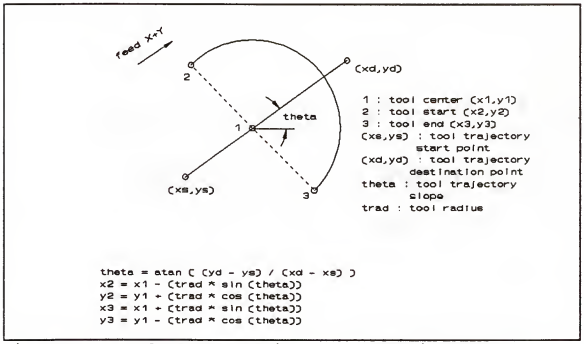

Figure 4.3 Tool representation for a feed in the X+Y direction

Figure 4.4 shows an instance of the tool moving in the +Y direction during a slotting cut. The two intersection points are denoted as p and q. The angles "start" and "end" depict the starting and ending angles of cut. These angles are used to calculate the lengths "dis1" and "dis2." The radial immersion is then calculated as a summation or difference of these two lengths. This value of radial immersion is usually expressed as a percent of the tool diameter. Radial immersion calculations in other feed directions can be determined in a similar fashion, once the start and end angles of cut have been established.

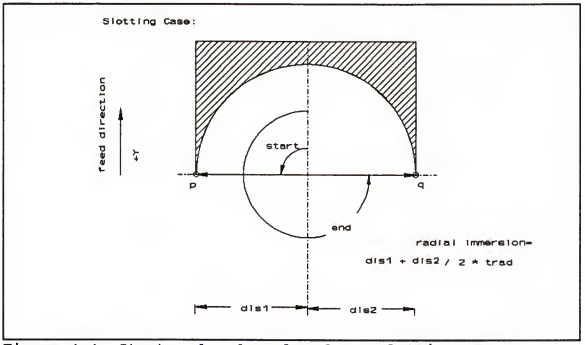

Figure 4.4 Start and end angles for a slotting case

Figure 4.5 shows a down milling cut at three quarters immersion in the +X direction. Figure 4.6 depicts an up milling cut at one quarter immersion in the -X direction. The

radial immersion in the latter case is the difference of "dis1" and "dis2", which is then expressed as a percent of the tool diameter.

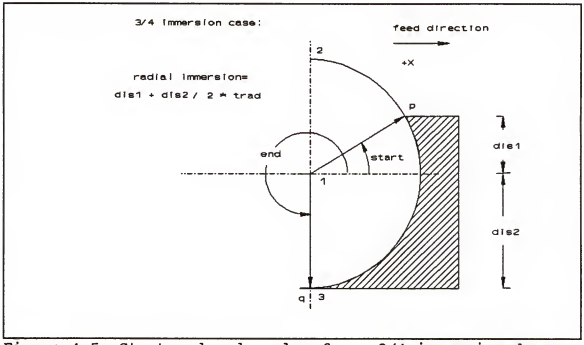

Figure 4.5 Start and end angles for a 3/4 immersion down milling case

It was mentioned above that in most instances the number of intersection points between the tool and removed area was two. Instances where the number of intersection points are zero and one are discarded as cases of zero immersion. However, during entry and exit conditions it is possible to have more than two intersection points. Figure 4.7 shows an instance where there are four points of intersection. This occurs during the exit stage on a slotting pass. The radial immersion in this case is calculated for points 1 and 2 at first. The procedure is then repeated for points 3 and 4. The

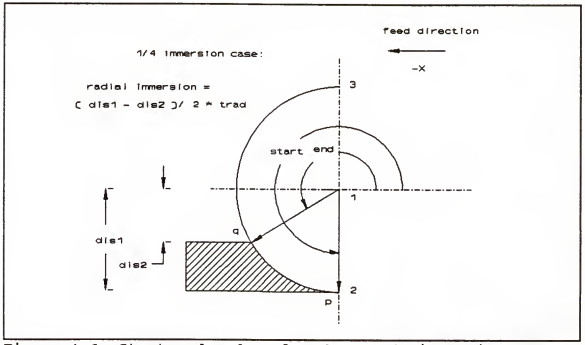

Figure 4.6 Start and end angles for a 1/4 immersion up-milling cut

total radial immersion is then expressed as a sum of these immersions.

Figure 4.8 shows instances where the number of intersection points is three. Both cases again occur during the exit stage of a slotting pass. In the first case, the valid intersection points are 1 and 2, while 1 and 3 are the desired points in the second case. It is necessary for the intersection detection algorithm to discard this redundant point and select the proper points. A possible solution would be to look at the number of intersection points the previous instant in the tool motion. In the first case the number of points would have been four while it would have been two in the second case. Schemes to calculate the radial immersion and direction of feed have been outlined above.

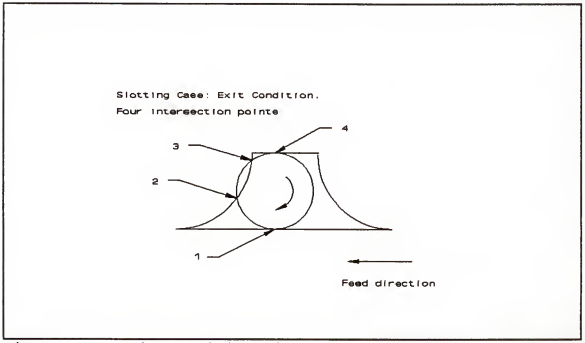

Figure 4.7 Exit Condition with four intersection points

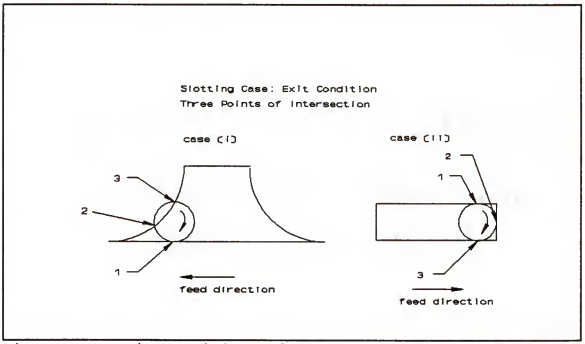

Figure 4.8 Exit conditions with three points of intersection

The next step is to calculate the axial depth of cut. Since the method is restricted to 2-D models the axial depth computations are simplified considerably. It becomes necessary to distinguish between the various depths by using the concept of cutting planes. The reference plane is taken as the top surface of the workpiece and has a z-value of zero assigned to it. Any positive z-values indicate a tool traversal in air. These z-values are read in from the NC file by the parser routine mentioned in Chapter 2. Figure 4.9 shows a simple case where a square blank needs to be machined in two passes, the first pass at a depth of -0.25 inches and the second at a depth of -0.6 inches. The axial depth for the first pass is -0.25 since the initial cutting plane is at the reference plane. The current cutting plane is now updated to be at a depth of -0.25 inches. Then the second pass has an axial depth of -0.35 inches, with the current cutting plane being further updated to a depth of -0.6 inches at the end of the pass. Figure 4.10 shows a situation where it would not be possible to apply the above scheme. The entire tool pass is subdivided into three stages A, B and C. The axial depth according to the above scheme would be calculated as -0.25, but from the figure it is obvious that the axial depths for portions A and C of the cut would be zero. Therefore the cutting plane scheme is restricted to cases where the workpiece has plane surfaces and the axial depths are increased in discrete steps. These could be overcome by using 3-D modelers. It should be noted here

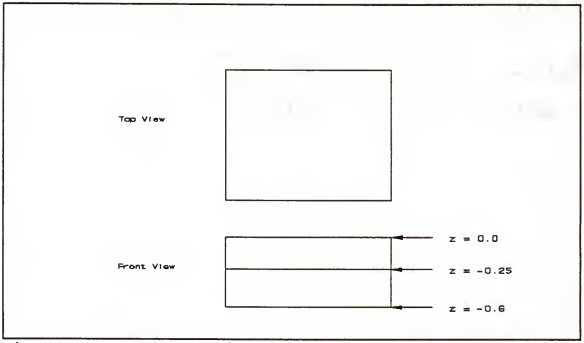

Figure 4.9 Representation of different cutting planes for axial depth calculations

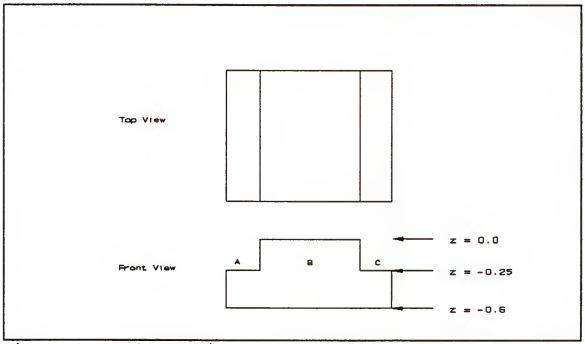

Figure 4.10 Shortcoming of current axial depth computation procedure

that the intent of the current work is to demonstrate the capabilities of a verification system and not to develop a modeler, several of which already exist. The developed system has been designed with a degree of flexibility, so that it can be used with few changes on 3-D models of the workpiece and tool representations.

The mode of milling is the last cutting parameter which needs to be determined to complete the first stage of the verification process. The three classifications that exist are up-milling, down-milling and slotting. The intersection points between the cutting tool representation and the removed area which were determined above can be used to classify the mode type. Slotting occurs at full radial immersion which can be identified by checking the value of the radial immersion calculated earlier. In this case, the value of the radial immersion (combination of the lengths dis1 and dis2) should be equal to the tool diameter. The other two types can be identified by utilizing the intersection points with the removed area. Figures 4.5 and 4.6 show two situations with the former depicting a down-milling cut at 3/4 immersion and the latter an up-milling cut at 1/4 immersion, with the feed directions being in the horizontal direction. The distinguishing feature between the two cases is that in up milling the chip thickness increases from zero at the beginning of the cut to maximum at the end, and in down milling the chip thickness decreases from maximum at the

beginning to zero at the end of the cut. A geometrical interpretation of this information needs to be developed to extract the mode type from the intersection points. In Figure 4.5 the ending angle of the cut is π radians, which corresponds to the case where the second intersection point q with the removed area is coincident with the end point 3 of the cutting tool representation. In Figure 4.6 the starting angle of cut is 0 radians, which represents a coincidence between the first intersection point p and the starting point 2 of the tool representation. In both the cases the coincidence condition will occur only at the point where the chip thickness drops to zero. A check for these two cases can be used to distinguish between up and down milling cuts.

It should be pointed out here that the above checks for the mode type would hold up for steady state cutting conditions where the number of intersection points between the tool representation and the removed area is two. Referring to Figure 4.7 which shows an exit condition during a slotting cut, the situation encountered here is one where the mode type is not an up-milling or down-milling case exclusively. The cut between points 1 and 2 is of the up-milling type while between 3 and 4 the mode of milling is down. As mentioned earlier, the radial immersion for this case is expressed as the sum of the individual radial immersions between 1 and 2, and 3 and 4. To establish the mode type, it is necessary to see which of these two individual immersions is the larger one and set the mode

for the entire cut the same as the one associated with this higher immersion.

Monitoring the Limits of Stability

This section describes the second stage of the verification process where the different cutting parameters determined in the first stage are compared against certain bounding values stored in a database. Figure 4.11 shows a model of the database that is used for the purposes of monitoring the cutting parameters. This is a three dimensional structure with the X and Y axes depicting the limit axial depths b_{cr} and the corresponding radial immersion respectively. The direction of feed is represented along the Z axis. The information in these databases represents the critical axial depths for a particular radial immersion, feed direction and mode of milling, above which there is an onset of chatter. It is necessary to characterize every single tool on every machine tool according to the above guidelines. The methodology used to construct these databases will now be provided in detail.

Zamudio (1991) has established well-defined techniques to characterize the various machine tools by using simulation methods as well as cutting tests. It has been shown by Tlusty et al. (1990) that for any radial immersion of the tool, there is an axial depth of cut above which chatter will occur. In

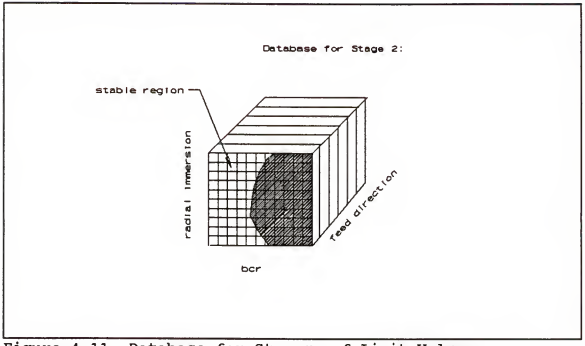

Figure 4.11 Database for Storage of Limit Values

general, as the radial immersion increases, the permissible axial depth of cut decreases. This permissible depth of cut depends on the real part of the oriented transfer function, the radial immersion and the direction of cut as expressed by Eqn. 1.13 in Chapter 1. The most reliable way to compute the chatter limits is to use the time domain simulation of the milling process. The simulation which is based on the regeneration of waviness model, currently includes the effects of process damping and other non-linearities such as the teeth jumping out of the cut. The critical parameters that are used in the simulation can be separated into four groups:

a) cutter geometry, which includes the tool diameter and the number of teeth.

b) system dynamics, which include the system modal parameters.

c) material properties, which are mainly the cutting stiffness of the material and the process damping coefficient.
d) cutting geometry, which includes the spindle speed, the axial and radial immersion, chip load, direction of feed, etc.

Time and frequency domain records of the cutting force and the vibration of the system are obtained from the simulation and can be examined to determine whether the system is stable or unstable. Figure 4.12 shows the time and frequency domain records of the vibration in the X direction for a 0.75 radial immersion cut with a six fluted end mill for both stable and unstable cases. Chatter is recognized by a large spectral line in the frequency domain which is not a harmonic of the spindle runout frequency. The severity of the vibration in the time domain plot can also be used to distinguish chatter.

The chatter limit for a given radial immersion can be determined by producing a composite peak-to peak diagram of the cutting force or the system vibration or both (Smith, 1987). The simulation program is run for a matrix of spindle speeds and axial immersions, and the peak-to-peak value of the time domain force and vibration signals are extracted and saved. The p-t-p values are plotted as shown in Figures 4.13 and 4.14 where the p-t-p vibration and force in the X direction are plotted versus spindle speed for lines of constant axial immersion. The axial immersion in this case is increased from 0.5 mm to 3.5 mm in steps of 0.125 mm. As the

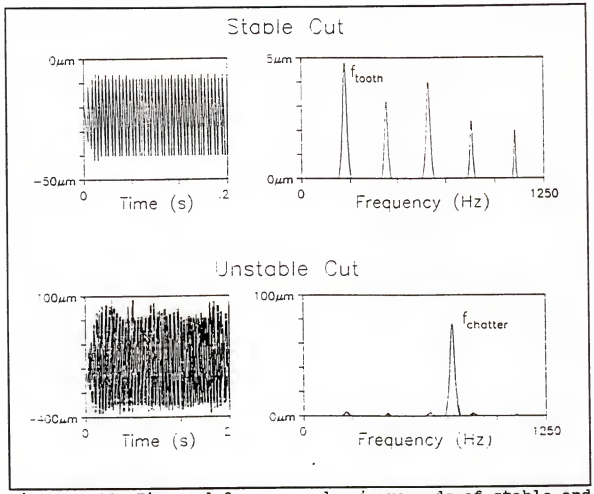

Figure 4.12 Time and frequency domain records of stable and unstable cuts produced by simulation

axial immersion is increased, some regions grow faster than others. The regions where the p-t-p values are relatively smaller are the stable regions, and the regions where they increase considerably faster are the unstable regions. It is possible to determine the chatter limit at a given spindle speed by finding the lowest axial immersion at which the p-t-p values start to increase rapidly.

The real parts of the Direct Transfer Functions (DTF) measured in the direction of table motion (denoted as X) and

perpendicular to the direction of table motion (denoted as Y) at the tip of a tool are shown in Figure 4.15. The machine used here was a Series 20 Sundstrand Omnimil. The particular tool used here was a four fluted high-speed steel end mill 25mm in diameter and of length 100mm. If the tool alone were involved in the dynamics, these transfer functions would be the same. However, in most practical cases, the structure of the machine also plays a role and the dynamics in the two directions may be quite different. Only one mode is needed to represent the X-direction dynamics, while two modes are required in the Y-direction as shown in Table 4.1.

Table 4.1 Dynamics for a 4-fluted 25 x 100 HSS end mill

Direction	k (N/m)	m (Kg)	ζ	f_n (Hz)
X	4.17×10^6	0.12	0.030	942
Y	1.02×10^7	0.30	0.015	928
Z	6.98×10^6	0.18	0.022	985

Based on this dynamic information, a number of simulations were run to evaluate stability for a range of immersions. This information was used to construct the database of limit values and is compiled in Table 4.2 for both up and down milling cases. It can be seen that the limit axial values have been determined for radial immersions which are incremented in steps of ten percent. The limit values at intermediate radial immersions can be obtained by fitting a

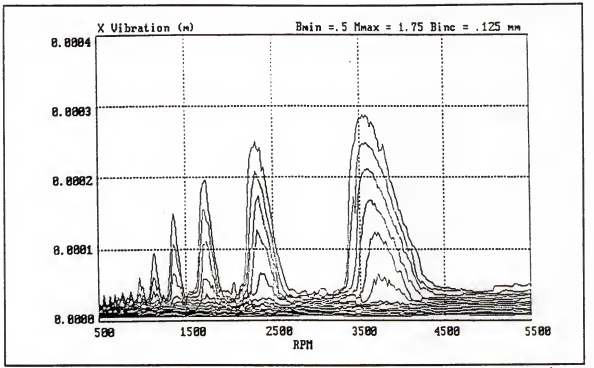

Figure 4.13 Composite p-t-p vibration diagram for slotting

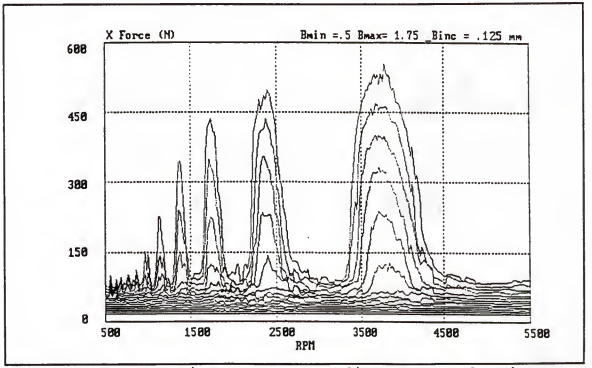

Figure 4.14 Composite p-t-p force diagram for slotting

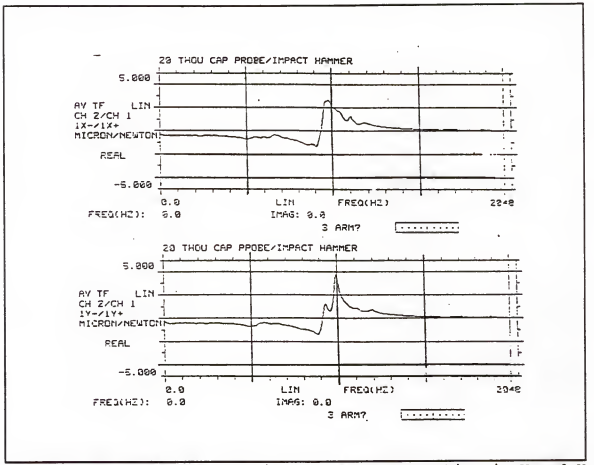

Figure 4.15 Real part of Direct Transfer Function in X and Y directions

curve through these points which produces results with a satisfactory level of accuracy.

The monitoring process consists of comparing the axial depths calculated in Stage 1 with the limits in the database for a particular cutting situation. It should be pointed out here that it is easier to correct the tool path using the axial depth than using the radial depths, since the former process does not involve a reprogramming of the entire tool path whereas the latter does. If the calculated axial depth exceeds the limit value, the current tool motion will result

Table 4.2 Permissible Depths of Cut for a 25mm by 100mm 4-Flute HSS End Mill Cutter on White-Sundstrand Machine (Zamudio, 1991)

			Axial Immersion for Four Cutting Dir. (mm)			
			+X	+X+Y	+Y	-X+Y
Radial Immersion	Up Milling	0.2	1.4	1.3	2.25	1.7
		0.3	0.9	1.05	1.7	1.5
		0.4	0.7	0.75	1.1	0.9
		0.5	0.6	0.6	0.7	0.7
		0.6	0.5	0.5	0.6	0.55
		0.7	0.4	0.45	0.4	0.45
		0.8	0.35	0.35	0.35	0.35
		0.9	0.3	0.3	0.3	0.3
		1.0	0.25	0.25	0.25	0.25
	Down Milling	1.0	0.25	0.25	0.25	0.25
		0.9	0.3	0.3	0.3	0.3
		0.8	0.35	0.35	0.35	0.35
		0.7	0.4	0.4	0.44	0.4
		0.6	0.5	0.5	0.48	0.45
		0.5	0.7	0.65	0.55	0.55
		0.4	1.0	0.7	0.65	0.7
		0.3	1.2	1.0	0.9	1.1
		0.2	1.6	1.8	1.3	2.1

in an unstable cut and should be repeated at a lower axial depth. The above process is repeated at every instant in the simulation, which makes it possible to identify all the regions of unstable machining. A status flag is set to either true or false depending on whether the cut is stable or unstable and passed on to the next stage of the system. The limit values used to create the above database have been validated by both cutting tests and simulation programs.

Reprogramming of the NC file

The third and final stage of the verification process concentrates on the corrective techniques that need to be undertaken in the event of chatter detection in the previous stage. As stated earlier, the representation breaks the tool motion into several small increments. It is possible to encounter chatter at several of these instances. It is thus left to the discretion of the programmer whether to repeat the entire tool motion at a lower axial depth or only those unstable regions when they are encountered. The current scheme repeats the entire tool motion at a lower depth.

It is convenient to reprogram with a lower axial depth, since the geometry of the tool path is not altered. Changes in radial depths would make the process more complicated since this would involve regressing back to the tool path generation modules to generate a new tool path with different geometry.

Thus the proposed scheme uses the axial depths as the variable. In the event of chatter detection the current scheme reduces the axial depth to half its original value and repeats the entire cut in two passes. Another viable alternative would be to reduce the axial depth to the limit value specified in the database and repeat the cut in the number of passes necessary to complete the machining cycle.

This optimization process modifies the NC file and writes in the new blocks of code when there is detection of chatter. The net result of this procedure is a new NC file with the same tool path geometry as before but at chatter free axial depths. This new NC file could be run through the verification system again as a check for stable machining. A sample listing of a few blocks of code from an NC file which resulted in chatter and the resulting corrected blocks from the verification system is provided below:

Original file:

```
N10 X+0.3750 Y+4.1250
N20 M03 S1000 F20.0
N30 G01 Z-1.125
N40 X4.1250
```

Optimized file:

```
N10 X+0.3750 Y+4.1250
N20 M03 S1000 F20.0
N30 G01 Z-1.125
N40 Z+1.0
```

```
N50 Z-0.5625
N60 X+4.1250
N70 Z+1.0
N80 X+0.3750
N90 Z-1.125
N100 X+4.1250
```

In the original NC file chatter occurs at an axial depth of -1.125 inches during the tool motion from X = 0.3750, Y = 4.1250 to X = 4.1250, Y = 4.1250. Block numbers 40 through 100 in the optimized file show the corrective procedures undertaken. The first step involves reducing the axial depth to half the value (-0.5625 inches). Block number 60 depicts the tool motion at this new depth. Blocks 70 and 80 represent the tool retraction in air and travel back to the start point, while 90 and 100 repeat the cutting motion at the full value of the depth (-1.125 inches).

This three stage corrective process makes up the nucleus of the verifier for chatter prediction. The representation consisting of the inputs and Boolean operations is a very important module which needs to communicate efficiently with the verifier to build up the overall verification system. Chapter 5 illustrates the application of the system on a spiral-in pocketing routine and also outlines the flowchart used in carrying out the various sequential operations.

CHAPTER 5
MODIFICATIONS IN THE REPRESENTATION SCHEME

This chapter illustrates the application of the verification system in a spiral pocketing routine. The specific case which is demonstrated here is the dynamic spiral-in pocketing routine as specified by Kshonze. In addition a flowchart of the algorithm used to carry out the various representation and verification techniques is provided. A sample listing of the output from the verification system is also shown. Some modifications in the pocket representations are also suggested to extend the use of this system to the spiral-out and zig-zag pocketing routines.

Dynamic Spiral-in Pocketing Routine

Kshonze (1989) has performed a comprehensive investigation on the tool path generation techniques used in some of the commercially available CAM packages such as Smartcam Ver. 4.5 by Point Control Co., and Expedite-3D by Numeridex. His studies have shown several limitations in these routines. Internal corners where the radial depth of cut changes drastically, are encountered in a majority of these routines. In addition for a given axial depth of cut, only one

radial immersion can be specified for all directions of feed. In the case of the commercially available linear routines, the tool removes material in only one direction and then retracts and traverses back in air to sink into the material and resume metal removal. Since the tool sinks into the material at the beginning of each pass, it would be necessary to have numerous pre-drilled holes, or a center-cutting end mill.

Dynamic pocketing routines have been developed by Kshonze which seek to resolve some of the above mentioned problems in the commercial routines. These routines have the ability to specify different radial depths of cut for different directions of motion, while still maintaining a stable cut throughout the machining process. Figure 5.1 shows the tool path generated by his dynamic spiral-in routine. The pocket chosen here is a rectangular one with no islands. The periphery of the pocket contains only line segments although arcs can also be included. There are two initial slotting passes around the periphery of the pocket (2-3-4-5-2 and 6-7-8-9-10) followed by the spiral-in motion. Note that the spiral motion has different radial immersions for the different feed directions. By introducing the machine dynamics and the mode of milling in the design of the tool path, the radial immersion can be increased in the more stable feed directions and decreased in the lesser stable directions. This scheme reduces the cutting time while still maintaining the same quality in machining.

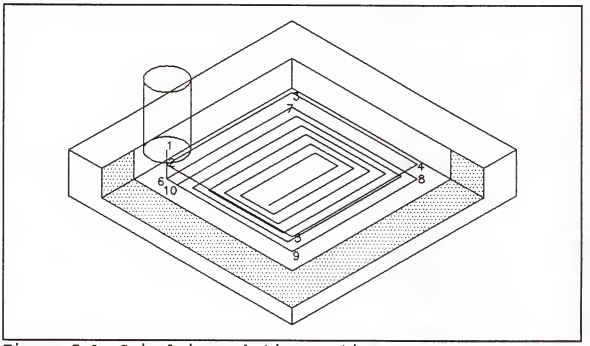

Figure 5.1 Spiral-in pocketing routine

The application of the verification system to predict chatter on the tool motions generated by the dynamic spiral-in routine is now demonstrated. The inputs to the verification system need to be modified for this specific cutting operation. The outer periphery of the pocket which is the starting geometry is now used as one of the inputs, with the other input being the tool swept area representation constructed from the NC program. The data file representing the pocket geometry consists of the start and end points of the various entities that make up the pocket profile. The only entities used in this case are line segments. The intersection points between every tool motion and the pocket geometry are determined to calculate the tool removed area and the updated area of the pocket to be machined on the next tool motion.

Figure 5.2 shows the result of the difference operation after the first tool motion on the first slotting pass. Figure 5.3 shows the tool swept area after the intersection operation on the same motion. The extraction of the cutter engagements along with the other cutting parameters, the monitoring of chatter and the reprogramming of the NC file is carried out using the techniques developed in Chapter 4. Figure 5.4 outlines the flowchart used in performing the above operations. The first phase consists of reading the inputs from the files, construction of the representation models and application of the Boolean operations. The verifier is comprised of the calculation of cutting parameters, along with the checking of these values against the limit values in the database and reprogramming of the NC file in the event of chatter detection.

Figure 5.5 shows a sample listing of the output at any instant in the simulation process. The calculated values of the cutting parameters and the status of the chatter flag are displayed at all instances. A color coding scheme is used to distinguish between the regions of stable and unstable machining. The current and destination tool positions are also listed as part of the output. In addition to this graphical output, the reprogrammed NC file is created as a secondary output. Blocks of code in the original file which were flagged as unstable tool trajectories are reprogrammed in the new file

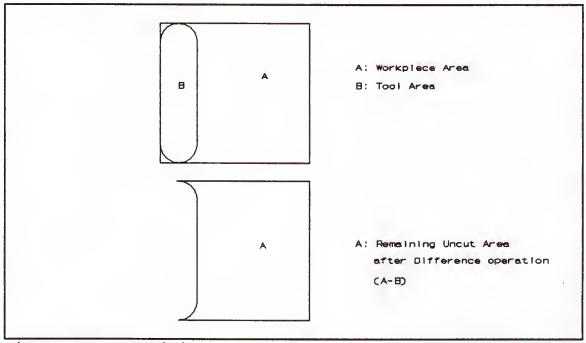

Figure 5.2 Remaining Uncut Area after Difference Operation

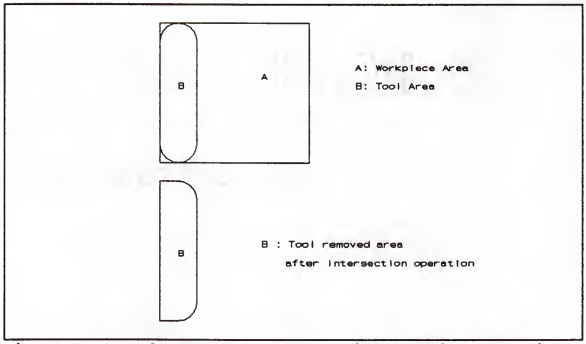

Figure 5.3 Tool removed area after intersection operation

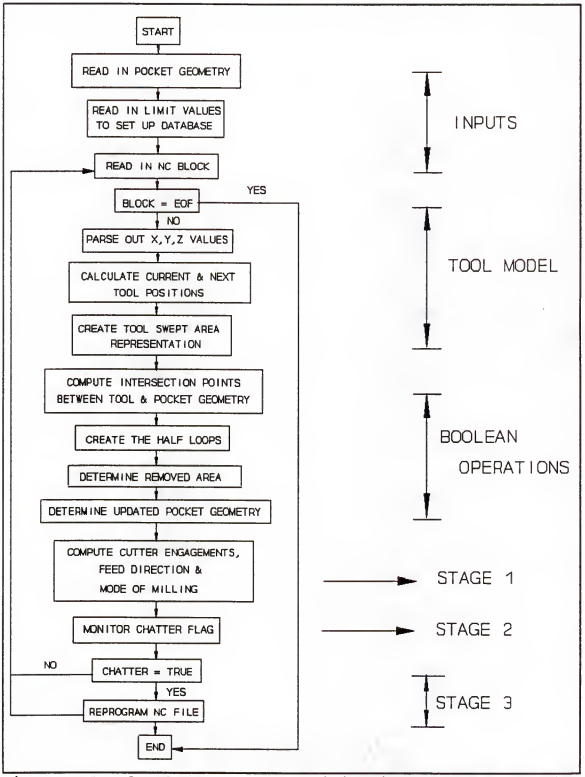

Figure 5.4 Flowchart for the verification system as applied to pocketing routines

at lower axial depths, while the stable blocks are copied unchanged into the new file.

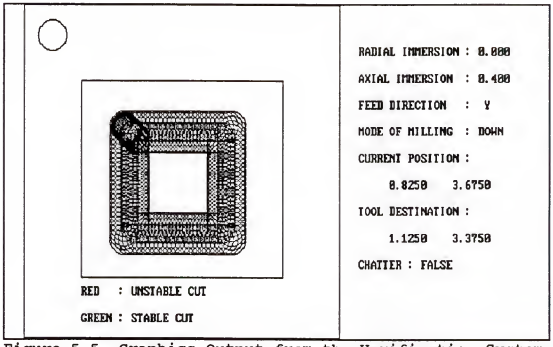

Figure 5.5 Graphics Output from the Verification System

The representation schemes outlined above store only the current model of the pocket geometry at all times. The Boolean difference operator is used to generate the remaining area to be machined at the end of every tool motion. However, no record is maintained of the starting profile of the pocket. There are certain drawbacks in this type of representation. The dynamic spiral-in routine involves two slotting passes at certain specified axial depths before the spiral-in motion is started. Figure 5.6 shows the pocket representation at the end of the first slotting pass generated using the current schemes. The dotted lines show the starting profile of the pocket constructed from the input data file. Since no record

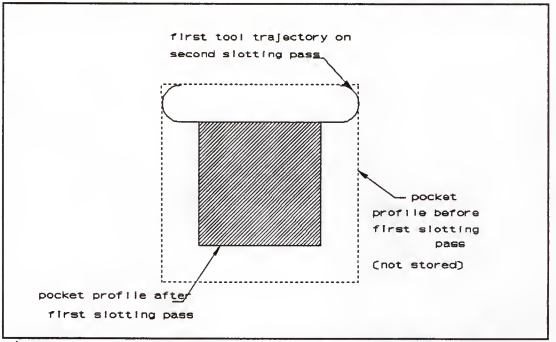

Figure 5.6 Pocket representation at the end of the slotting pass

is maintained of this starting geometry at the end of the first slotting pass, it becomes impossible for the intersection detection algorithm to determine any intersection points on subsequent tool motions during the second slotting pass. In addition, most of the commercially available pocketing routines like the spiral-out routine use tool paths which start from the center of the pocket and proceed outward in a spiral fashion. In this kind of situation it is also impossible to obtain intersections between the tool trajectory representation and the pocket geometry. Thus, it becomes necessary to modify the representation schemes for the pocket geometry to eliminate these shortcomings. The required modifications are detailed in the section below.

The Modified Representation Scheme

The representation schemes used up to this point demand a pre-knowledge of the geometry of the pocket to determine the intersections between the tool trajectory and the pocket geometry. However, as mentioned in the previous section, this technique cannot be applied in specific cases like the commercially available spiral-out routines. In order to broaden the scope of application of the verification system, a modification of the current representation scheme is presented here. The starting geometry of the pocket does not serve as a useful input in the spiral-out routine and can be eliminated. It will then become necessary to create the pocket profiles entirely from the NC tool paths. The pocket geometry will need to be updated at the end of every tool motion. This scheme eliminates the data file containing the starting pocket geometry as an input, thus leaving the programmed NC file as the sole input to the verification system. The application of the modified scheme on the dynamic spiral-in routine is shown below.

In the case of the dynamic spiral-in routine, the spiral motion is preceded by the initial slotting passes. By eliminating the starting pocket geometry as an input, the need arises to generate this profile from the tool swept areas on the first slotting pass. In contrast to the prior schemes which used the difference operator to generate the area yet to

be machined, the modified scheme uses the union operator on the slotting passes to obtain the area that has already been machined. The difference operator cannot be used in the present scheme since the pocket geometry is not available from which to subtract the tool swept areas. The pocket geometry needs to be built by creating the union of these swept areas.

Figure 5.7a shows the tool swept area representations (denoted as areas 1 and 2) of the first two motions on the first slotting pass. The intersection detection algorithms are now called to detect all legal intersection points between these two areas. All intersections between coincident lines and between arcs with the same center are discarded. The points p and q in Figure 5.7b are the two legal intersection points returned by the detection algorithms. The loop splitting routine is now used to separate the areas 1 and 2 at the points of intersection. In Figure 5.7b "loop1" and "loop2" are the two half loops associated with area 1 obtained as a result of the split, while "loop3" and "loop4" define the area 2.

The union operator is used to generate an updated model of the pocket geometry, which represents the total area that has been machined at the end of the second tool motion on the first slotting pass. A linking routine is used to connect the two loops "loop1" and "loop4" to obtain the total machined area as shown in Figure 5.8a. The removed area for the second tool motion shown in Figure 5.8b can be obtained by using the

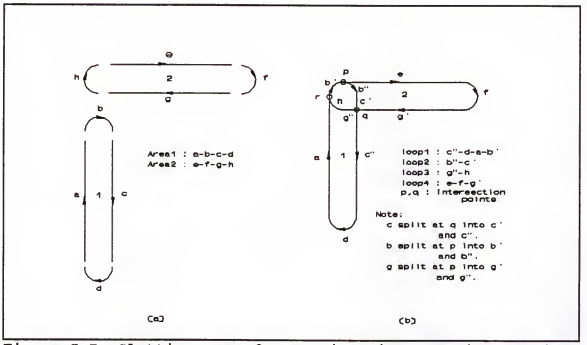

Figure 5.7 Slotting pass for a spiral-in pocketing routine a) first two tool motions; b) Loop splitting operation

intersection operator which involves a linking of "loop2" and the reverse of "loop4." The reverse of "loop4" is obtained by reversing the direction of the entities that define it and relinking them. The reverse of "loop4" in this case is g'-f-e. The removed area is necessary for the cutting parameter extraction process which was described in Chapter 4.

Successive uses of the union operator during subsequent tool motions on the first slotting pass define the pocket geometry. Figure 5.9a shows the tool swept area representation D on the last tool motion during the first slotting pass. The intersection detection algorithm returns four intersection points p, q, r and s. The union operator is used to generate two profiles referred to as "node1" and "node2" in Figure

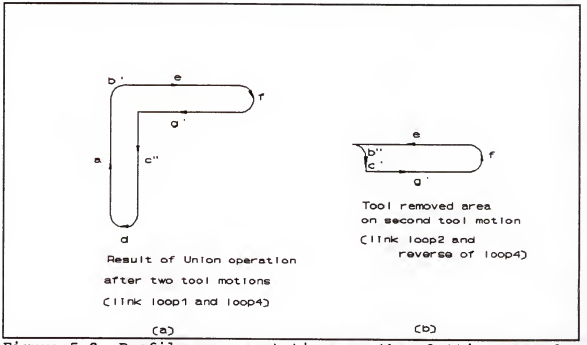

Figure 5.8 Profile representations on the slotting pass for a spiral-in pocketing routine
a) Machined area; b) Removed area

5.9b. The profile "node1" represents the outer periphery of the pocket (or the starting geometry), while the area contained inside profile "node2" is the area that is yet to be machined.

On subsequent tool motions, the intersection detection routine checks for intersections between the tool swept area representations and the profiles "node1" and "node2." The second slotting pass in the dynamic spiral-in routine produces intersections with "node1", while the spiral-in motion will generate intersections with "node2." Once "node2" (which represents the area left to be machined) has been obtained, only the difference operator is required to generate the updated models of the pocket. The union operator is no longer

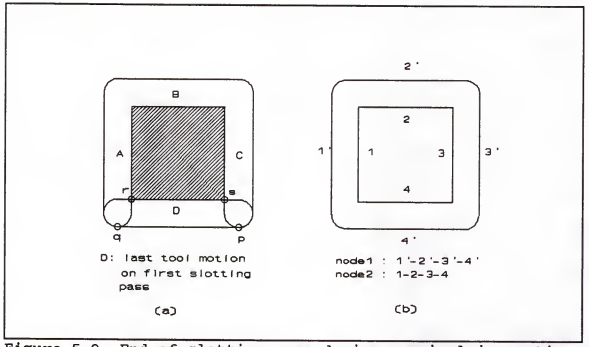

Figure 5.9 End of slotting pass during a spiral-in routine
a) Last tool motion; b) Profile representations

used now, since an updated model of the pocket geometry is available from which to subtract the tool swept areas on subsequent tool motions. Figure 5.10a shows the two intersection points p and q between the tool swept area and "node2", detected during the first tool motion on the spiral pass. Figure 5.10b illustrates the application of the difference operator to obtain the remaining unmachined area. Note that the updating process now involves the generation of the remaining area that needs to be machined, while the union operation on the slotting passes involved the creation of the area that has already been machined. The removed area on all these tool motions is calculated as before by using the intersection operator.

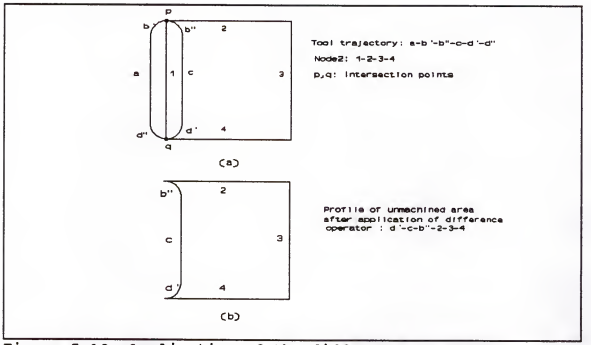

Figure 5.10 Application of the difference operator
a) Intersection detection; b) Unmachined area

The profile "node2" is updated at the end of every tool motion during the spiral pass. However, "node1" is not updated after the end of the initial slotting passes since all subsequent tool motions involve intersections with "node2" for this type of pocketing routine. A record of "node1" is always maintained in order to perform calculations during the final finishing pass which involves intersections between the tool swept area and "node1."

The profiles "node1" and "node2" are also represented by structures similar to the ones illustrated in Chapter 2. The structure template for "node1" and "node2" is defined as follows:

```
struct node {

                struct node *prev;
```

```
        struct node *next;
        struct edgedefine *start;
    }
```

For the present case, node1 -> prev = node2 -> next = NULL, node1 -> next = node2 and node2 -> prev = node1. Since the pocket geometry for this example is a simplified model, there are only two profiles "node1" and "node2" to be dealt with. In the case of pockets with complicated geometries and also with islands, there will be a necessity to maintain a record of more than two profiles to check for potential intersections. In such cases, the pointers "prev" and "next" are very useful in proceeding in a systematic fashion through a list of these nodes. The structure "edgedefine" contains the geometric information about the starting entity in the profile and also the addresses of its next and previous entities. Its structure template is the same as the one described in Chapter 2.

It has been shown that the modified representation scheme can be applied to the dynamic spiral-in routine without causing any changes in the cutting parameters extraction and corrective processes. The next chapter illustrates the application of the verification system with the modified schemes in spiral-out, linear and zig-zag pocketing routines with and without islands.

CHAPTER 6
APPLICATIONS IN COMMERCIAL POCKETING ROUTINES

The previous chapter highlighted the application of the modified verification system on the dynamic spiral-in routine, which was a specialized routine created to overcome some of the problems encountered in the commercially available routines. However to justify the scope of application of the developed system, its use in these commercial routines needs to be demonstrated. The following sections illustrate the application of the system in the spiral-out and zig-zag routines with and without the presence of internal islands.

Description of the Pocketing Routines

This section describes the tool paths generated by some of the pocketing routines on Smartcam Ver. 4.5 by Point Control Co. The pocketing routines in this software package can be classified into three major categories:

i) Spiral-out: In this routine as shown in Figure 6.1, the tool initially sinks into the material near the center of the pocket and from there it traverses in a spiral fashion outward, generating the periphery of the pocket with the last pass. This type of pocket encounters numerous internal

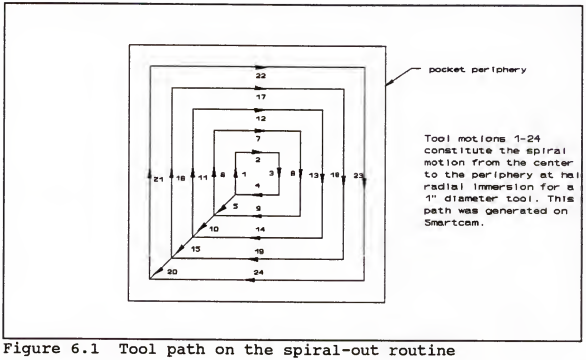

Figure 6.1 Tool path on the spiral-out routine

corners. Additionally, the radial depth of cut changes drastically in the tool path, and increases when the periphery of the spiral is enlarged.

ii) Zig-Zag: In this type of pocket as shown in Figure 6.2, the tool is first sunk into the workpiece at one of the corners. The tool then moves back and forth across the pocket removing material in both modes of milling (up and down), with the last pass creating the far side of the pocket. For a given axial depth of cut, since only one radial immersion can be specified for both modes of milling, it is necessary to use the lower stable radial immersion. Since both modes of milling are utilized and the smaller radial immersion is used throughout the pocket, the efficiency of the routine is decreased.

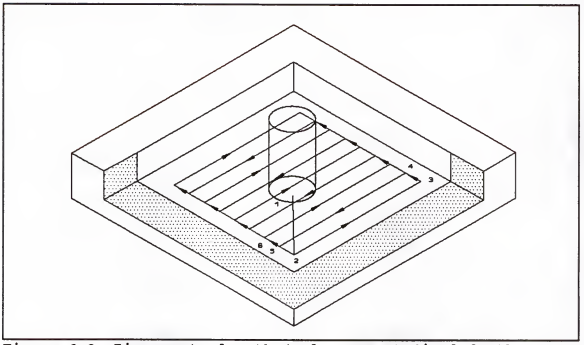

Figure 6.2 Zig-zag tool path (only pass at final depth of cut is shown)

iii) Linear: This routine as shown in Figure 6.3 allows only one mode of milling, either up or down. The tool removes material in one direction and then retracts and traverses back in air to sink into the material and resume metal removal. Since the tool sinks into the material at the beginning of each pass, the preferred alternative includes an initial slotting of the periphery (2-3-4-5-2) before the linear motion. The tool traversal in air is significantly higher in this routine than in the other ones, thus resulting in increased machining time and reduced productivity. The application of the verification system in a linear routine is not demonstrated, since the techniques used are similar to the ones used for the zig-zag routine.

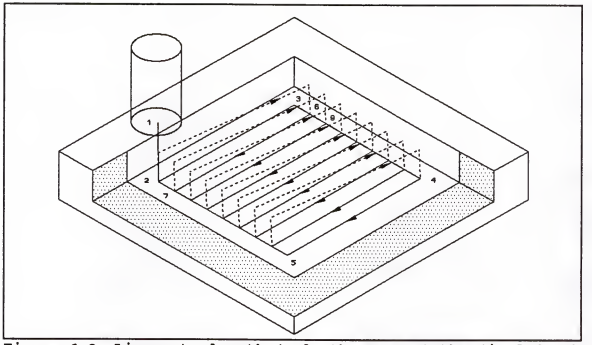

Figure 6.3 Linear tool path (only the pass at the final depth of cut is shown)

Spiral-out Pockets without Islands

This section illustrates the application of the verification system in the case of a rectangular pocket with no internal islands, machined using the spiral-out algorithm generated on Smartcam. The tool path as shown in Figure 6.1 consists of an initial slotting pass followed by the spiral motion. The modified representation scheme is used on the slotting pass, where the union operator is applied to generate the pocket profile. In Figure 6.4a the four tool motions on the slotting pass are shown starting at the center of the pocket. Figure 6.4b shows the pocket profile "node1" generated as a result of successive applications of the union operator at the end of the slotting pass. The profile "node1"

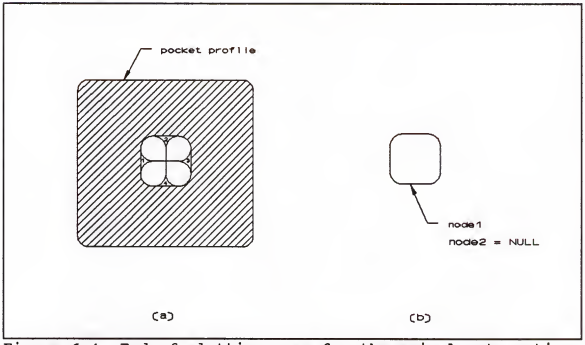

Figure 6.4 End of slotting pass for the spiral-out routine
a) Tool motions; b) Profile representation

represents the area that has already been machined whereas in the case of the dynamic spiral-in routine, "node1" represented the outer periphery of the pocket. The profile "node2" is set to NULL since there is no unmachined area inside the profile "node1." In the event of a second slotting pass at a lower depth, the above procedure can be repeated once again to generate "node1" at the lower axial depth.

In the case of the dynamic spiral-in routine, the spiral motion at the end of the slotting pass generated intersections between the tool swept areas and "node2." The pocket profile on the spiral motion was generated by an updating process whereby the difference operator was applied to obtain the remaining area that had to be machined. In the case of the spiral-out routine, since the tool path proceeds from the

center outward towards the periphery of the pocket, the spiral motion is considered as an expansion process which continuously generates the area that has already been machined. Thus the profile "node1" is updated at the end of each tool motion via the union operator to obtain the machined area.

It should be pointed here that in the case of the dynamic spiral-in routine the difference operator is used to generate the updated pocket profiles on the spiral motion, while the union operator is used in the case of the spiral-out routine. Thus there arises a necessity to distinguish between the type of pocketing operation that is used. Figure 6.5 shows the first tool motion after the slotting pass during the spiral-out pocketing operation. The point P represents the tool destination point for the current tool motion. This point lies outside the profile "node1" for this case, whereas for the spiral-in case the point P would lie within the profile "node1." Thus the winding number calculations described in Chapter 3 can be used to distinguish the inside-outside relationship of this point with respect to the closed area "node1". The winding number takes on a value of 0 for the spiral-out case, and a value of -1 for the spiral-in path.

Figure 6.6a shows the first four tool motions on the spiral pass for the spiral-out routine carried out at half radial immersion. Figure 6.6b shows the updated profile "node1" at the end of the fourth tool motion on the spiral

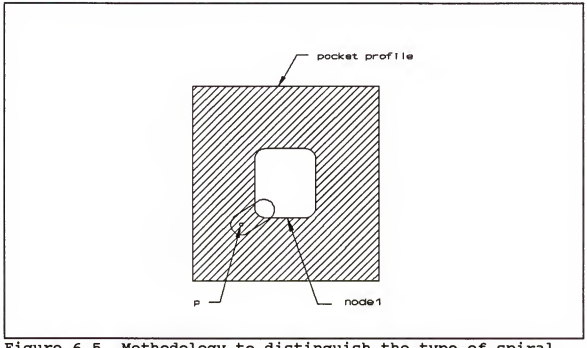

Figure 6.5 Methodology to distinguish the type of spiral pocketing operation used

pass. The removed area generated during the spiral motion is different than the corresponding area generated in the case of the spiral-in routine. In the spiral-in routine illustrated in Chapter 5 which was a down-milling cut, the material being removed is to the right of the cutting tool when viewed along the feed direction. The present spiral-out routine involves machining in the up-milling mode, where the material being removed is to the left of the cutting tool. Thus, the half loops which need to be linked together to generate the removed area are different in this case. Figure 6.7a shows the swept area representation 1 for a tool motion in the +Y direction during the spiral motion. The points p and q are the points of intersection between the tool swept area and "node1." Figure 6.7b shows the removed area and Figure 6.7c shows the updated

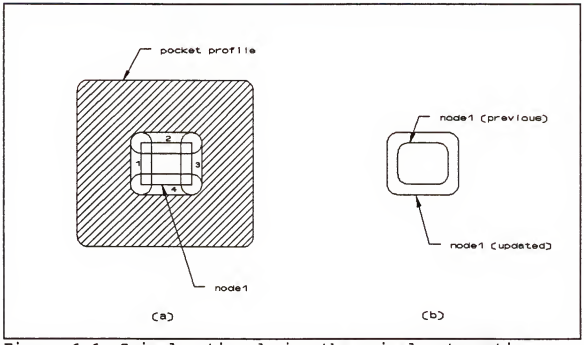

Figure 6.6 Spiral motion during the spiral-out routine
a) Tool motions; b) Profile representations

profile "node1" at the end of the current tool motion.

The radial immersion calculations detect the increases in the radial depth from half to full immersion at the internal corners. Currently, the present corrective techniques recommend reprogramming the current tool motion at a lower axial depth to combat the occurrence of chatter at corners. Tsai et al. (1991) have come up with some preliminary techniques whereby chatter at the corners can be controlled by modifying the tool path at the circle tool path segment so that the engagement angle remains constant. This can be done by adding additional tool path segments at the corner. This strategy could be potentially considered as an alternate corrective technique to optimize the tool path generation at internal corners.

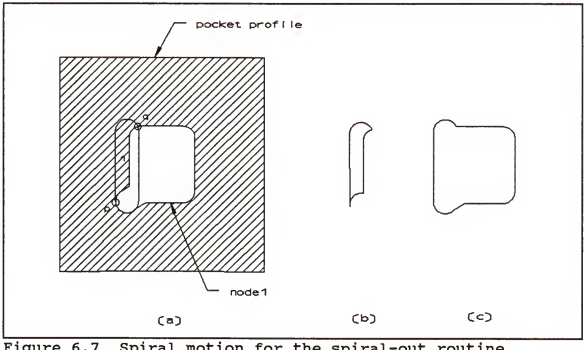

Figure 6.7 Spiral motion for the spiral-out routine
a) Intersection detection; b) Removed area;
c) Updated pocket area

Spiral-out Pockets with Islands

The presence of internal islands within the pocket profile needs to be considered when dealing with some of the commonly used pocketing operations. The present section illustrates the application of the verification system on a rectangular pocket with one internal island. The first case presented contains an island with a rectangular profile, and the second case contains a triangular shaped island. The tool paths used here are once again spiral-out paths at half radial immersion and generated on Smartcam.

Figure 6.8a shows the tool motions on the first slotting pass around the island periphery for a rectangular pocket with

a rectangular island. Successive applications of the union operator generate the profiles "node1" and "node2" at the end of the slotting pass, shown in Figure 6.8b. The profile "node1" as before represents the area that has been machined, while "node2" in this case represents the island profile. A record of "node2" is maintained at all times to ensure that any finishing passes at a later stage around the island periphery can be handled by the intersection detection algorithms. The spiral motion which follows the slotting pass results in an expansion of the profile "node1", thus providing an updated record of the machined area.

At the end of the spiral motion which generates the outer pocket periphery, the tool path generation algorithm on Smartcam includes a finishing pass around the island periphery. It should be pointed out here that during the spiral motion which was an up-milling cut, the material to be removed was to the left of the cutting tool when viewed along the feed direction. This resulted in a spiral motion with a clockwise orientation. However during the finishing pass around the island periphery, in order to maintain the up-milling mode of cut, the direction of the tool path needs to be reversed and proceeds in a counter clockwise fashion around the island profile. This condition needs to be programmed into the verification system so that the appropriate half loops can be linked together to generate the removed area. This finishing pass around the island profile generates

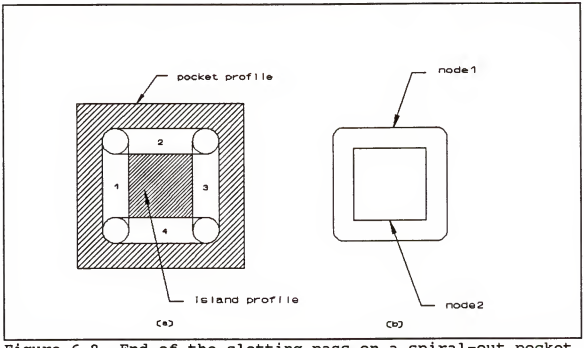

Figure 6.8 End of the slotting pass on a spiral-out pocket with a rectangular island
a) Tool motions; b) Profile representations

intersections between the tool swept area representation and "node2", thus justifying the need to keep a record of "node2" at the end of the slotting pass. Figure 6.9a shows the outer pocket profile ("node1") and the island profile ("node2") at the end of the spiral motion. Figure 6.9b shows the four tool motions in the counter clockwise direction on the finishing pass around the island profile. It should be pointed out that the radial immersion on this finishing pass is less than the programmed nominal half immersion for the spiral motion and is around 16 percent.

The second case illustrated deals with a rectangular pocket containing a triangular island. The spiral-out tool path for this pocket is a little different from the one

generated for the rectangular island, specifically on the slotting pass thus causing some check conditions to be programmed into the system. Figure 6.10 shows the tool path generated on Smartcam for this pocket. The first three tool motions on the slotting pass 1-2-3 generate a closed area as shown in Figure 6.11a. In the previous examples the generation of a closed area on the slotting pass indicated the end of the pass and resulted in the creation of the profiles "node1" and "node2." However in the present case although a closed area is obtained at the end of tool motion 3, this is not sufficient to generate the island profile ("node2"). Hence the next few tool motions need to be taken into account before a decision is made as to whether there is sufficient information to generate the two profiles. Figure 6.11b shows the machined area ("node1") at the end of tool motion 6 on the slotting pass. Figure 6.12a depicts the tool swept area representation for tool motion 7 on the same slotting pass. The points p, q, r and s are the four intersection points between "node1" and the tool swept area. A closed area is obtained once again and it is now possible to generate "node2" which represents the island profile by linking the appropriate half loops. Figure 6.12b shows the two profiles "node1" and "node2" obtained at the end of the tool motion number 7 on the slotting pass. Successive tool motions generate intersections with "node1" which is updated at each step. The slotting pass is followed by the spiral motion which is the same as the one in the case

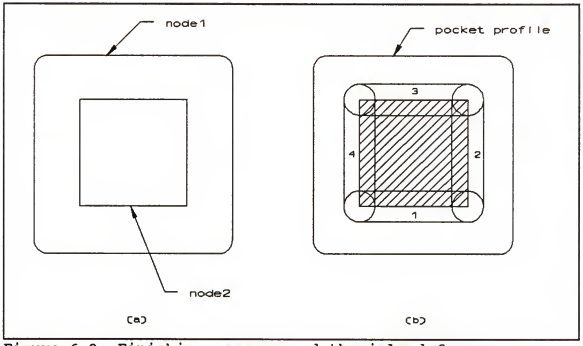

Figure 6.9 Finishing pass around the island for a spiral- out pocket with a rectangular island a) Profiles before the finishing pass; b) Reversal of tool motions on the finishing pass

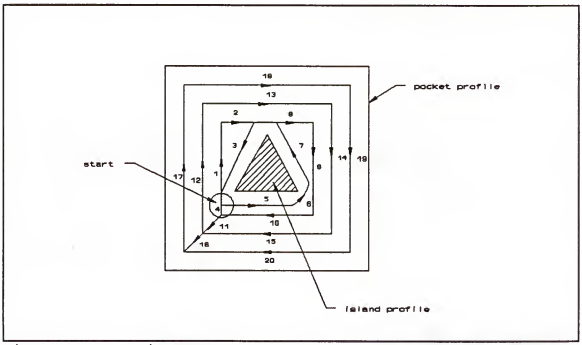

Figure 6.10 Spiral-out tool path for a rectangular pocket with a triangular island generated on Smartcam

of the pocket with the rectangular island. Although no finishing pass around the island profile is used in this case, such situations can be adequately dealt with, since a record of "node2" is always maintained.

The anomalies encountered in the above case with the triangular island could have been eliminated if the slotting pass around the island was simplified. Figure 6.13a shows a scheme which restricts the slotting pass around the island profile to only three tool motions. These tool motions are similar to the motions on the slotting pass generated for the pocket with the rectangular island. Figure 6.13b shows the two profiles "node1" and "node2" at the end of the slotting pass. A closed area is generated at the end of the third tool motion on the slotting pass, thus providing sufficient information to generate the profiles. However, in such an instance it maybe efficient to replace the spiral motion by a zig-zag motion, since the slotting passes around the island have already been completed. This case is dealt with in more detail in the next section, where pockets with similar triangular islands are chosen for illustration.

Zig-zag Pockets with and without Islands

The second type of commercially available pocketing routine which is chosen for illustrating the various schemes of the verification system is the zig-zag routine. The tool

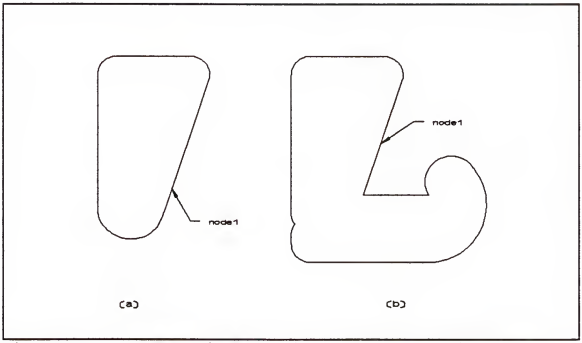

Figure 6.11 Intermediate profiles on the slotting pass for a spiral-out routine with a triangular island a) After 3 tool motions; b) After 6 motions

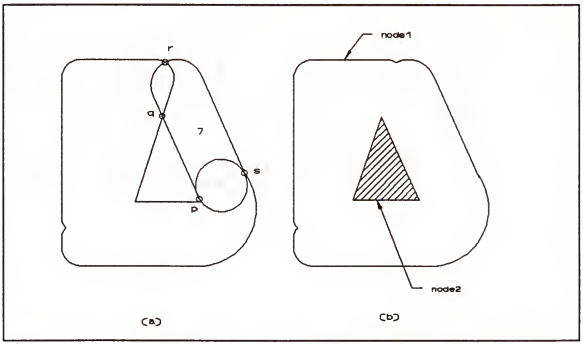

Figure 6.12 End of slotting pass during the spiral-out routine with a triangular island a) Intersection detection; b) Profiles

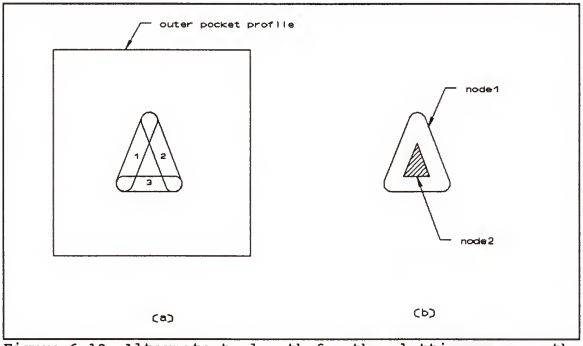

Figure 6.13 Alternate tool path for the slotting pass on the spiral-out pocket with a triangular island
a) Tool motions; b) Profile representations

path for machining a rectangular pocket without any internal islands was shown in Figure 6.2. Both up and down milling modes are encountered on the tool path in this type of routine. For the tool path shown in Figure 6.2 the radial immersion chosen was half immersion. In this type of routine there is no initial slotting pass around the periphery of the pocket. The only profile that is updated here is "node1", since there is no island profile ("node2") whose record needs to be maintained. The profile "node1" once again represents the area that has already been machined and is progressively updated at the end of every tool motion by use of the union operator. Figure 6.14a shows the first two tool motions on the zig-zag path. The points p and q are the two intersection

points returned by the intersection detection algorithm. Figure 6.14b shows the removed area for the second tool motion and the updated profile "node1" at the end of the two tool motions. The changes in the mode of milling from up to down and vice versa need to be monitored so that the appropriate half loops can be linked to generate the tool swept area and the updated profile.

Figure 6.15 presents the updated profile node1 at the end of the entire zig-zag pass. The areas of unremoved material which are left behind as a result of the cutting tool curvature are shown as hatched regions on the sides of the pocket. The tool path generation algorithm on Smartcam includes a finishing pass around the entire periphery of the pocket so that these areas of unremoved material can be machined. Figure 6.16 shows the first tool motion on this finishing pass. The points p, q, r, s, t, u and v are the multiple points of intersection returned by the intersection detection algorithm. These multiple intersection points make it impossible for the loop splitting algorithms to generate the half loops to create the removed area and the updated profile "node1." Hence an alternate solution needs to be pursued when a situation of this kind is encountered. Figures 6.17a and 6.17b shows the approach taken to solve this type of problem. The present tool motion is subdivided into several small sections and the intersection detection process is repeated for each of these smaller tool motions. Figure 6.17a

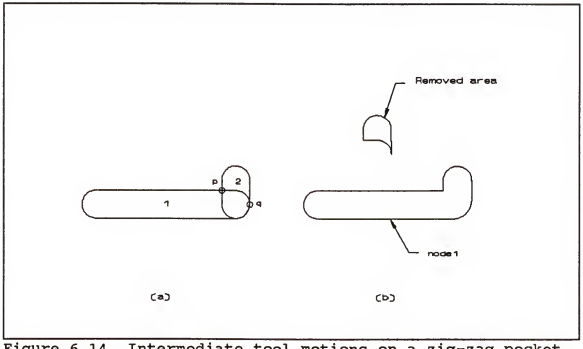

Figure 6.14 Intermediate tool motions on a zig-zag pocket without islands
a) Intersection detection; b) Removed area and updated machined area

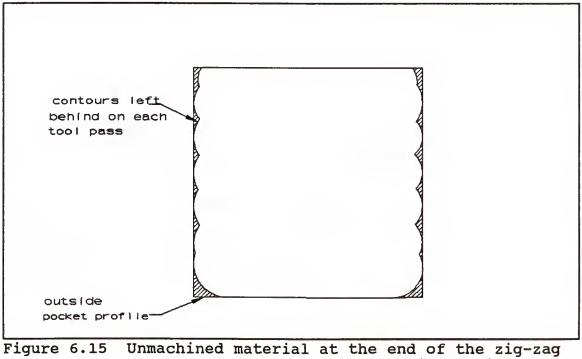

Figure 6.15 Unmachined material at the end of the zig-zag pass on a rectangular pocket with no island

shows the two intersection points p and q on the first subdivided tool motion. The generation of the tool swept area and updated profile "node1" is performed as before with no complications, since there are only two points of intersection. The second subdivided tool motion on this finishing pass is shown in Figure 6.17b, where again only two points of intersection are determined. This subdivision process is repeated until the tool destination point on the original full tool motion is reached.

Another case that is illustrated is the zig-zag pocketing routine on a rectangular pocket with a triangular island. Figure 6.18 shows the profile "node1" built up by use of the union operator at an intermediate stage of the tool path generation process. The tool swept area representation of the next tool motion is also shown in the figure. The points p, q, r and s are the four intersection points between the tool swept area and "node1". A closed profile is created by this last tool motion which makes it possible to generate "node2" which represents the island profile. Figure 6.19 shows the two profiles "node1" and "node2" at the end of the current tool motion. The tool motions on the finishing pass need to be subdivided as before to take into account the multiple intersection points that are generated by the intersection detection algorithm.

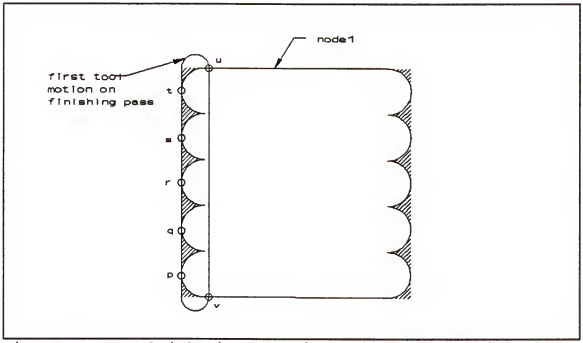

Figure 6.16 Multiple intersection points on the finishing pass of a zig-zag pocketing routine on a rectangular pocket with no island

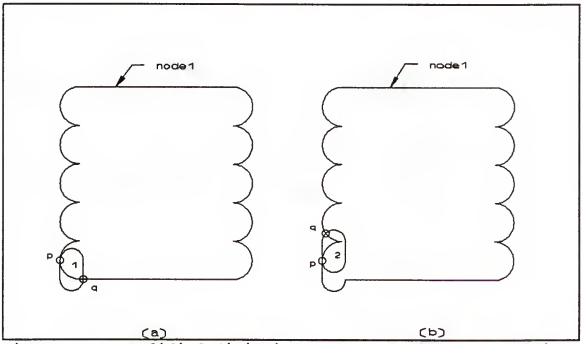

Figure 6.17 Modified finishing pass of a zig-zag pocketing routine on a rectangular pocket with no islands a) first tool motion; b) second tool motion

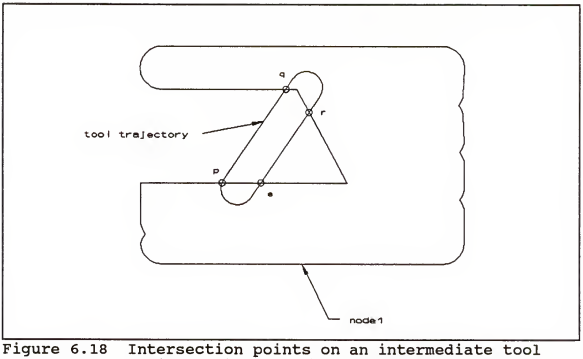

Figure 6.18 Intersection points on an intermediate tool motion during the zig-zag pocketing routine on a rectangular pocket with a triangular island

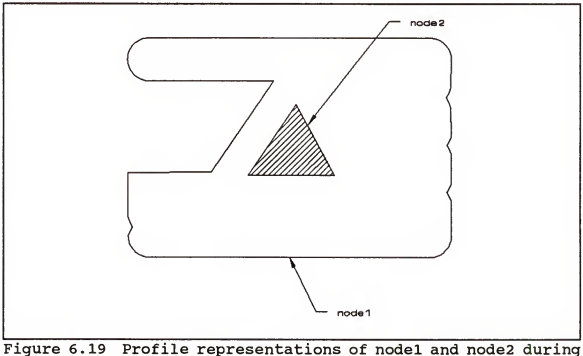

Figure 6.19 Profile representations of node1 and node2 during the zig-zag pocketing routine on a rectangular pocket with a triangular island

A Specialized Pocketing Routine

The last case which is illustrated in this chapter is a specialized pocketing routine where the tool path is generated from the algorithms developed by Cheng (1991). The tool path used is a spiral-in motion preceded by an initial slotting pass. Figure 6.20 shows the pocket along with the generated tool path. It is seen that after the slotting pass the spiral motion is restricted to only the rectangular portion of the pocket, since the remaining area has been completely machined during the slotting pass. Another feature that needs to be pointed out is that all circular motions in the tool path are approximated by a series of linear motions in the tool path generation routines. Figure 6.21a shows the last tool motion on the slotting pass. The points p, q, r and s are the four intersection points returned by the intersection detection algorithm. Figure 6.21b shows the two profiles "node1" and "node2" generated from the half loops created at these points. The profile "node1" represents the outside periphery of the pocket, while "node2" represents the area that is yet to be machined. The profile "node2" is updated at the end of every tool motion on the spiral pass by use of the difference operator.

Figure 6.22a shows a unique situation encountered during the last tool motion on the spiral pass. It is seen that the area to be machined is completely encompassed by the tool

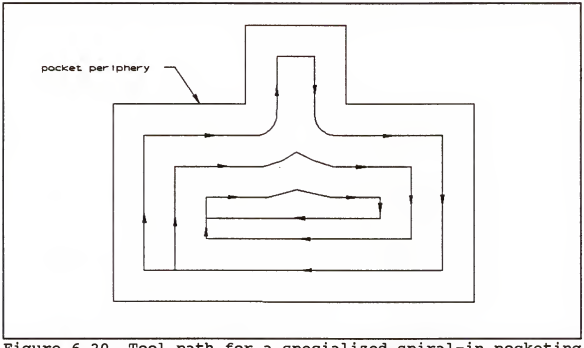

Figure 6.20 Tool path for a specialized spiral-in pocketing routine

swept area representation. This makes it impossible to detect any intersection points between the two areas. Figure 6.22b shows the approach taken to deal with such a situation. The current tool motion is subdivided into smaller sections (two in the present case), similar to the subdivision process used during the finishing pass on the zig-zag pockets. The points p1 and q1 in Figure 6.22b represent the two intersection points between the first subdivided tool swept area and the unmachined area, while p2 and q2 represent the second set of intersection points. It should be pointed out here that the profile "node2" is not updated at the end of the first subdivided tool motion, since it will not be possible to generate any intersection points during the second subdivided tool motion. The profile node2 gets updated only at the end of

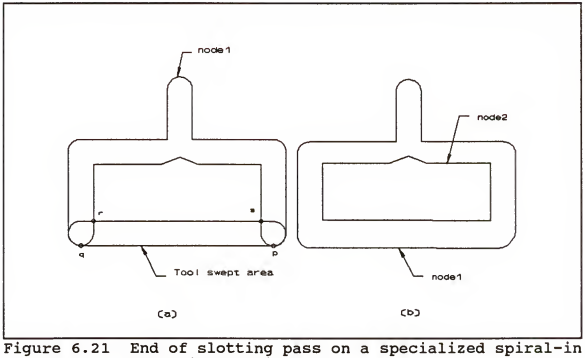

Figure 6.21 End of slotting pass on a specialized spiral-in pocketing routine
a) Intersection detection; b) Profile representations

the second tool motion.

The application of the verification system in some of the commonly used pocketing operations has been demonstrated in this chapter. The pocket profiles along with the island profiles were restricted to simple geometries consisting of lines and arcs, although complicated geometries with multiple internal islands can also be used with a few changes in the object representation schemes. It has also been shown that the modified representation scheme extends the scope of application of the system to a variety of commercial and specialized pocketing routines, while reducing the actual number of inputs to the system.

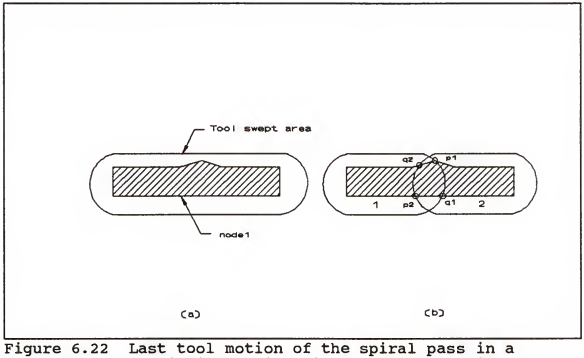

Figure 6.22 Last tool motion of the spiral pass in a specialized spiral-in pocketing routine
a) Profile enclosed by tool swept area;
b) Subdivision process

CHAPTER 7
CONCLUSIONS AND RECOMMENDATIONS

The applications in commercial and specialized pocketing routines of the verification system which were presented in Chapter 5 and Chapter 6, demonstrate the successful working of the chatter prediction schemes and the corrective techniques in some of the commonly used machining operations. The use of computer verification of the machining process greatly reduces the need for several iterations of tool path testing as well as minimizing the machine time costs.

The geometries of the pockets and islands chosen were restricted to simple two dimensional geometries comprised of lines and arcs. In order to extend the scope of application of the system, more complicated geometries with multiple internal islands need to be considered. This will necessitate upgrades in the intersection detection schemes to deal with unique situations that may arise as a result of the inclusion of these geometries.

Another aspect that needs to be investigated is upgrades in the object representation schemes. The modified representation scheme presented in Chapter 5 seems to work adequately for simple geometries. The simple linked list method of representation may not be an efficient method for

pockets with multiple internal islands, since no record of the relationship between the various islands and the pocket profile is maintained. An alternate form of representation that needs to be investigated is the binary tree representation used by Cheng (1991), which is designed to handle most of the complicated geometries.

The extraction of the cutting parameters was successfully demonstrated in the various pocketing routines. Radial immersion calculations for any given feed direction were performed by determining intersections between the cutting tool models and the tool swept area. Some work still needs to be done in the area of radial immersion calculations for tool motions involving circular interpolation. In addition, the use of 3-D models for object representation along with the inclusion of solid modeling needs to be considered for cases that may include changes in the axial depth on a single tool motion. The cases considered were limited to a fixed axial depth along any given tool motion.

The corrective technique that has been implemented presently is the reprogramming of the current tool motion at lower immersions in the event of chatter detection. The ability to redesign the entire tool path in cases where chatter is detected on a significant number of tool motions, and also modification of the tool path at internal corners are features that need to be implemented as optimization tools. Although the main criterion around which this present system

was developed was the avoidance of chatter, the required databases could be prepared for criteria of surface finish, accuracy, cutting force etc.

Although the system in its present state is restricted to simple applications, the verification techniques presented in this research represent a significant step towards the future development of an efficient chatter prediction system. Most of the past work in the area of verification software has concentrated on aspects such as gouging, detection of excess material and tool and fixture collisions. Although the importance of the role of chatter in machining operations is widely recognized, no effort has been made to include schemes to detect chatter in any of the commercial verification systems. However, the availability of the databases of limit values of stability along with the rapid advances in the area of geometric modeling in NC machining applications, should open the doors for future work in the area of off-line chatter detection systems.

APPENDIX
SUPPORT ALGORITHMS FOR THE BOOLEAN OPERATIONS

This section provides a description of the various support routines that are necessary for the execution of the various Boolean operations. The intersection detection algorithms that are referred to in the earlier chapters are outlined in detail in this appendix. These algorithms include routines to detect the intersections between two lines, between two arcs and between a line and an arc. Other secondary routines described include techniques used to determine whether a specified point lies on a given line or arc and whether a specified point lies between the start and end points of a line.

Intersection Between Two Lines

This routine determines the point of intersection (xint,yint) between two lines. Let (x1,y1) and (x2,y2) be the start and end points respectively of the first line and (x3,y3) and (x4,y4) the start and end points of the second line. The slopes of the two lines are given by,

$$m1 = (y2 - y1) / (x2 - x1) \quad (A.1)$$

$$m2 = (y4 - y3) / (x4 - x3) \quad (A.2)$$

If (x2 - x1) = 0 then,

$$xint = x1 \tag{A.3}$$

$$yint = y3 + m2 * (xint - x3) \tag{A.4}$$

If (x4 - x3) = 0 then,

$$xint = x3 \tag{A.5}$$

$$yint = y1 + m1 * (xint - x1) \tag{A.6}$$

There are no intersection points between coincident lines (m1 = m2 = 0) or parallel lines (m1 = m2). For all other cases,

$$xint = (y3 - y1 + m1*x1 - m2*x3) / (m1 - m2) \tag{A.7}$$

$$yint = m1 * (xint - x1) + y1 \tag{A.8}$$

Intersection Between a Line and an Arc

This routine calculates the intersection point (xint,yint) between a line and an arc. Let (x1,y1) and (x2,y2) be the start and end points of the line and (x3,y3), (x4,y4) and (x5,y5) the center, start and end points of the arc. For the purposes of calculation the arc is represented by a circle drawn with the same center and radius. This generates two intersection points (xint1,yint1) and (xint2,yint2) between the circle and line. The valid intersection point can be obtained by checking to see which of these two points is contained within the original arc.

The radius r1 of the arc is given by,

$$r1 = sqrt ((x4 - x3)** 2 + (y4 - y3)** 2) \tag{A.9}$$

In the case of a vertical line (x1 = x2), the distance d is given by,

```
d = (x3 - x1)                                          (A.10)
```

If (d = 0),

```
xint1 = x1                                             (A.11)
yint1 = y3 + r1                                        (A.12)
xint2 = x1                                             (A.13)
yint2 = y3 - r1                                        (A.14)
```

If (d <= r1),

```
xint1 = x1                                             (A.15)
yint1 = y3 + sqrt (r1**2 - d**2)                       (A.16)
xint2 = x1                                             (A.17)
yint2 = y3 - sqrt (r1**2 - d**2)                       (A.18)
```

In the case of a non-vertical line,

```
m1 = (y2 - y1) / (x2 - x1)                             (A.19)
```

Any point (x6,y6) on the line is represented by,

```
x6 = (m1**2*y3 + m1*x3 - m1*x1 + y1) / (1 + m1**2)     (A.20)
y6 = -m1*y4 + m1*y3 + x3                               (A.21)
```

The distance d1 between this point and the center of the arc is given by,

```
d1 = sqrt ((x3 - x6)**2 + (y3 - y6)**2)                (A.22)
```

If (d1 > r1) no intersection is possible. Otherwise,

```
d2 = sqrt ((r1**2 - d**2))                             (A.23)
```

Unit vectors i and j are defined by,

```
i = (x2 - x1) / sqrt ((x2 - x1)**2 + (y2 - y1)**2)
                                                       (A.24)
```

```
j = (y2 - y1) / sqrt ((x2 - x1)**2 + (y2 - y1)**2)
```
(A.25)

```
xint1 = x6 + d2 * i
```
(A.26)

```
yint1 = y6 + d2 * j
```
(A.27)

```
xint2 = x6 - d2 * i
```
(A.28)

```
yint2 = y6 - d2 * j
```
(A.29)

Intersection Between Two Arcs

This routine calculates the intersection point (xint,yint) between two arcs. Let (x1,y1), (x2,y2) and (x3,y3) be the center, start and end points of the first arc and (x4,y4), (x5,y5) and (x6,y6) the center, start and end points of the second arc. The radii r1 and r2 of the two arcs are given by,

```
r1 = sqrt ((x2 - x1)**2 + (y2 - y1)**2)
```
(A.30)

```
r2 = sqrt ((x5 - x4)**2 + (y5 - y4)**2)
```
(A.31)

The distance d1 in Figure A.1 is given by,

```
d1 = sqrt ((x4 - x1)**2 + (y4 - y1)**2)
```
(A.32)

If (r1 + r2) < d1 or (r1 - r2) > d1 or (x1 = x4 and y1 = y4), then no intersection is possible. Otherwise the angles $\phi 1$ and $\phi 2$ in Figure A.1 are given by,

```
φ1 = acos ( (r2**2 - r1**2 - d1**2) / (2 * r1 * d1) )
```
(A.33)

```
φ2 = acos ( (x4 - x1) / d1 )
```
(A.34)

If y4 < y1 then,

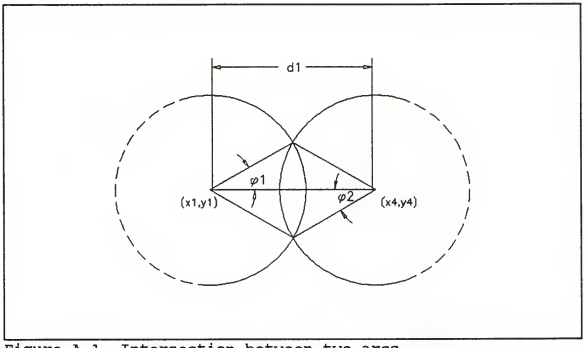

Figure A.1 Intersection between two arcs

$$\phi 2 = (2 * \pi) - \phi 2 \tag{A.35}$$

$$xint1 = x1 + r1 * \cos(\phi 2 + \phi 1) \tag{A.36}$$

$$yint1 = y1 + r1 * \sin(\phi 2 + \phi 1) \tag{A.37}$$

$$xint2 = x1 + r1 * \cos(\phi 2 - \phi 1) \tag{A.38}$$

$$yint2 = y1 + r1 * \sin(\phi 2 - \phi 1) \tag{A.39}$$

Check for Containment of a Point on a Line

This routine checks whether a point (xp,yp) lies on a line with start point (x1,y1) and end point (x2,y2). The slope m1 of the line is given by,

$$m1 = (y2 - y1) / (x2 - x1) \tag{A.40}$$

Let,

$$d1 = (yp - y1) \tag{A.41}$$

$$d2 = m1 * (xp - x1) \quad \text{(A.42)}$$

If d1 = d2, the point (xp,yp) lies on the given line.

Check for Containment of a Point on an Arc

This function checks whether a point (xp,yp) lies on an arc with (x1,y1) and (x2,y2) as its start and end points. The slope m1 of the line drawn between these start and end points is given by,

$$m1 = (y2 - y1) / (x2 - x1) \quad \text{(A.43)}$$

Let,

$$dx = 0.5 \quad \text{(A.44)}$$

$$dy = 0.75 \quad \text{(A.45)}$$

Depending on the orientation of the arc and its position in the four quadrants,

$$x3 = x1 \ (+ \text{ or } -) \ dx \quad \text{(A.46)}$$

$$y3 = y1 \ (+ \text{ or } -) \ dy \quad \text{(A.47)}$$

For a vertical line between (x1,y1) and (x2,y2) let,

$$temp1 = (x3 - x1) \quad \text{(A.48)}$$

$$temp2 = (xp - x1) \quad \text{(A.49)}$$

If (temp1 > 0) and (temp2 >= 0), or if (temp1 < 0) and (temp2 <= 0), the point is contained within the arc.

For a non-vertical line let,

$$temp3 = y3 - y1 \quad \text{(A.50)}$$

$$temp4 = m1 * (x3 - x1) \quad \text{(A.51)}$$

$$temp5 = yp - y1 \quad \text{(A.52)}$$

$$temp6 = m1 * (xp - x1) \quad (A.53)$$

If (temp3 - temp4) < 0 and (temp5 - temp6) > 0, or if (temp3 - temp4) > 0 and (temp5 - temp6) < 0, the point is contained within the arc. Note that these calculations are performed for an arc with a clockwise orientation.

Check for Containment of a Point between Two Points

This function checks whether a point (xp,yp) lies between the start and end points (x1,y1) and (x2,y2) of a line. Let the distances d1, d2 and d3 be given by,

$$d1 = sqrt ((yp - y1)**2 + (xp - x1)**2) \quad (A.54)$$

$$d2 = sqrt ((yp - y2)**2 + (xp - x2)**2) \quad (A.55)$$

$$d3 = sqrt ((y2 - y1)**2 + (x2 - x1)**2) \quad (A.56)$$

If (d1 + d2) = d3, then the specified point lies between the start and end points.

REFERENCES

Altintas, Y., Yellowley, I., "The Identification of Radial Width and Axial Width in Peripheral Milling," International Journal of Machine Tools Manufact., Vol. 27/3, pp. 367-381, 1987.

Anderson, R.O., "Detecting and Eliminating Collisions in NC Machining," Computer Aided Design, Vol. 10, No. 4, pp. 231-237, 1978.

Chappel, I.T., "The Use of Vectors to Simulate Material Removed by Numerically Controlled Milling," Computer Aided Design, Vol. 15, No. 3, pp. 156-158, May 1983.

Cheng, E.H., "Dynamic Pocketing Routine for any Two and a Half Dimensional Pocket with Islands," Proposal of Topic for Ph.D. Dissertation, University of Florida, 1991.

Frishdal, R., Cheng, K.P., Duncan, D., Zucker, W., "Numerically Controlled Part Program Verification System," Proceedings of Conference on CAD/CAM Technology in Mechanical Engineering, MIT Press, pp. 236-256, March 1982.

General Electric Company, "Milling Handbook of High Efficiency Metal Cutting,", n.p., 1980.

Jerard, R.B., Hauck, K., Drysdale, R.L., "Simulation of NC Machining of Sculptured Surfaces," 15th International Symposium on Automotive Technology and Automation (ISATA), paper no. 86057, Flims, Switzerland, Oct. 6-10, 1986.

Kshonze, S., "Dynamic Pocketing Routines for NC Machining," Masters Thesis, University of Florida, 1989.

Mantyla, M., An Introduction to Solid Modeling, Computer Science Press, Rockville, Maryland, 1988.

Melkote, S.N., Kendall, L.A., "End Mill Cutter Engagement Determination from CNC Tool Trajectories for Planar Surface Generation," Proc. NAMRC, SME, pp. 152-159, 17th May 1989.

Metcut Research Associates Inc., "Machining Data Handbook," Cincinnati, Ohio 1972.

Mortenson, M., Geometric Modeling, John Wiley and Sons, 1985.

Oliver, J.H., Goodman, G.D., "Color Graphic Verification of NC Milling Programs for Sculptured Surfaces," 10th Annual Automotive Computer Graphics Conference and Exposition, Engineering Society of Detroit, Dec. 1985.

Persson, H., "NC Machining of Arbitrarily Shaped Pockets," Computer Aided Design, Vol. 10, No. 3, pp. 169-174, 1978.

Requicha, A.A.G., Voelcker, H.B., "Solid Modeling: A Historical Summary and Contemporary Assessment," IEEE Computer Graphics and Applications, Vol. 2, No. 2, pp. 9-22, pp. 9-22, March 1982.

Smith, S., "Automatic Selection of Optimum Spindle Speed in High-Speed Milling," Ph.D. Dissertation, University of Florida, 1987.

Smith, S., Tlusty, J., "NC Programming for Quality in Milling," Proc. NAMRC, SME, pp. 279-286, 16th May, 1988.

Sungertekin, U.A., Voelcker, H.B., "Graphical Simulation and Automatic Verification of NC Machining Programs," Proc. of IEEE International Conference on Robotics and Automation, pp. 156-165, San Fransisco, April 7-10, 1986.

Takata, S., Tsai, M.D., Inui, M., Sata, T., "A Cutting Simulation System for Machinability Evaluation Using a Workpiece Model," Annals of CIRP, Vol. 38/1, pp. 417-420, 1989.

Tlusty, J., Smith, S., "Forced Vibration, Chatter, Accuracy in High Speed Milling," Proc. NAMRC 13, SME, May 1985.

Tlusty, J., "Machine Dynamics," Handbook of High Speed Machining Technology, R.I. King ed., Chapman and Hall, New York, 1985.

Tlusty, J., Ismail, F., "Special Aspects of Chatter in Milling," Journal of Vibrations, Stress and Rel. in Design, Vol. 105, pp. 24-32, 1983.

Tlusty, J., Smith, S., Zamudio, C., "New NC Routines for Quality in Milling," Annals of the CIRP, Vol. 39/1/1990, pp. 517-521, 1990.

Tsai, M.D., Takata, S., Inui, M., Sata, T., "Operation Planning Based on Cutting Process Models," Annals of the CIRP, Vol. 40/1/1991, pp. 95-98, 1991.

Van Hook, T., "Real Time Shaded NC Milling Display," ACM SIGGRAPH, 20, 4, pp. 15-20, 1986.

Voelcker, H.B., Hunt, W.A., "Role of Solid Modeling in Machining - Process Modeling and NC Verification," SAE Technical Paper 810195, 1981.

Wang, W.P., Wang, K.K., "Real Time Verification of Multiaxis NC Programs with Raster Graphics," IEEE Proc. of 1986 Conference on Robotics and Automation, San Fransisco, pp. 166-171, April 7-10, 1986.

Zamudio, C., "Theoretical and Practical Background for the Characterization of CNC Machining Centers," Topic of Ph.D. Dissertation, University of Florida, 1991.

BIOGRAPHICAL SKETCH

The author was born on December 9, 1964 in Madras, India. After obtaining a Bachelor of Science degree in mechanical engineering from Anna University at Madras in 1986, he moved to Gainesville, Florida, to pursue his graduate studies. He earned a Master of Science degree in industrial engineering in 1988 from the University of Florida at Gainesville. After completion of his doctoral degree he plans to pursue a career in research and development in the areas of computer-aided design and manufacturing.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jiri Tlusty, Chairman
Graduate Research Professor of
Mechanical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Carl Crane
Assistant Professor of
Mechanical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

John K. Schueller
Associate Professor of
Mechanical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Kevin Scott Smith
Assistant Professor of
Mechanical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Sencer Yeralan
Associate Professor of
Industrial and
Systems Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August, 1992

Winfred M. Phillips
Dean, College of Engineering

Madelyn M. Lockhart
Dean, Graduate School